PRINCIPI MATEMATICI DELLA ROBOTICA II

Frontiere della Robotica Intelligente

Robot Umanoidi · Modelli VLA · World Models Fisici

Imitazione a Basso Costo · Calcolo Neuromorfico

Principi matematici avanzati e architetture della nuova generazione

PARTE I

Robot Umanoidi a Scala Reale: dal Laboratorio all'Industria

Nota all'Edizione Unificata

Il presente volume unifica in un unico corpus organico la monografia principale "Principi Matematici della Robotica II — Frontiere della Robotica Intelligente" e il relativo supplemento. I contenuti supplementari sono stati integrati nel punto logicamente più appropriato di ciascuna parte, eliminando la separazione artificiale tra testo principale e appendice separata. I capitoli di nuova integrazione (2.7, 3.7, 7.4, 9.7, 12.7, 12.8, 15.6, 16.5, 20.5, 21.5) espandono il trattamento originale con: modelli di forza e ammettenza adattiva, raccordo HZD–RL, architetture VLA di terza generazione, world model per corpi deformabili, manipolazione bimanuale e data augmentation in SE(3), pianificazione differenziabile, PINN per l'identificazione online, composizione di barrier functions, calcolo VLA su hardware neuromorfico, framework software open-source e una tavola delle principali formule supplementari. Le appendici E e F raccolgono rispettivamente i riferimenti ai tool open-source e le formule chiave. La bibliografia è stata unificata con i riferimenti aggiuntivi del supplemento.

Capitolo 1 — Architetture dei Sistemi Umanoidi di Nuova Generazione

La transizione che separa i prototipi da laboratorio dai sistemi operativi industriali costituisce uno degli snodi più delicati nella storia dell'ingegneria robotica. Negli ultimi due anni questo varco è stato attraversato da almeno quattro piattaforme che presentano caratteristiche profondamente diverse fra loro, eppure convergenti verso un medesimo obiettivo: operare in ambienti progettati per esseri umani, eseguire compiti manipolativi di media complessità e farlo con un'affidabilità che giustifichi la presenza di tali macchine nelle catene produttive reali.

Figure 02, sviluppato da Apptronik con sede ad Austin, è un umanoide di circa 1,67 m e 75 kg la cui architettura attuativa si basa sugli attuatori elastici in serie (Series Elastic Actuators, SEA): tra l'albero motore e il giunto è interposto un elemento elastico di rigidezza calibrata k, il quale funge da sensore di forza — misurando la deformazione δ si risale alla coppia $\tau = k\delta$ — e da filtro meccanico degli impatti. Questa scelta, apparentemente retrograda in un'epoca in cui i sistemi idraulici ad alta densità energetica dominano la scena dei robot ad alte prestazioni, si rivela particolarmente opportuna per le applicazioni manifatturiere: la cedevolezza passiva riduce la probabilità di danneggiare oggetti fragili e compensa parzialmente gli errori di modello durante l'interazione fisica con l'ambiente.

Tesla Optimus, giunto con la seconda generazione alla produzione pre serie, adotta invece attuatori lineari a vite a ricircolo di sfere azionati da motori brushless a magneti permanenti ad alta densità di coppia. Il sistema sensoriale incorpora un'unità di misura inerziale ridondante, sensori di forza/coppia ai polsi e lo stesso hardware di visione sviluppato per i veicoli a guida autonoma, con la differenza sostanziale che il sistema di elaborazione — lo stesso SoC progettato internamente da Tesla — deve ora gestire in tempo reale la fusione sensoriale, la pianificazione del movimento e il controllo, tutto su un unico substrato computazionale di circa 144 TOPS.

Boston Dynamics ha compiuto nel 2024 una svolta radicale: il proprio umanoide Atlas, già celebre per acrobazie straordinarie nella versione idraulica, è stato interamente riprogettato in versione elettrica. La nuova architettura mantiene la struttura cinematica bipede ma sostituisce il sistema oleodinamico con attuatori rotativi ad alta efficienza. La ragione di questa transizione è in larga misura economica — la manutenzione dei sistemi idraulici in contesti industriali comporta costi operativi considerevoli — ma comporta vantaggi tecnici nel dominio del controllo: l'assenza di ritardi idraulici e la risposta più lineare degli attuatori elettrici facilitano la progettazione di controllori a banda larga.

Agility Robotics, con la piattaforma Digit v4, ha percorso una via ancora differente: il sistema è espressamente concepito per la logistica in ambienti strutturati come magazzini e centri di distribuzione. La cinematica delle gambe, ispirata alla morfologia degli uccelli cursori, consente una locomozione efficiente su superfici piane con un consumo energetico inferiore rispetto ai bipedi a coscia dritta, ma limita la capacità di adattamento a terreni irregolari. Le braccia, dotate di sette gradi di libertà ciascuna, permettono di afferrare, trasportare e depositare carichi fino a 16 kg. Dal punto di vista del controllo, Digit opera con un'architettura a tre livelli: un pianificatore di alto livello fondato su ottimizzazione convessa, un controllore di postura di medio livello che gestisce l'equilibrio e la ridondanza cinematica, e un regolatore articolare di basso livello ad anello chiuso sulla coppia.

1.1 Parametri Dinamici e Modelli di Corpo Rigido

Indipendentemente dall'architettura specifica, ogni robot umanoide è riconducibile al formalismo del corpo rigido multi-link. Lo spazio delle configurazioni di un sistema con n giunti è la varietà $Q = (S^1)^r \times \mathbb{R}^p$ dove r è il numero di giunti rotoidali e p quello dei prismatici. A questo spazio articolare si aggiunge, per un sistema free-floating, il gruppo di configurazione del corpo di base: lo stato completo è $(T_b, q) \in SE(3) \times Q$, con T_b che descrive la posa del torso rispetto al frame mondo.

Le equazioni del moto in forma Lagrangiana per un sistema con n gradi di libertà articolari e corpo di base floating si scrivono definendo il vettore di

stato generalizzato η = (p_b⊤, ω_b⊤, q⊤)⊤ ∈ ℝ^(6+n). Le equazioni del moto prendono la forma:

$$M(\eta)\,\ddot{\eta} \;+\; C(\eta,\,\dot{\eta})\,\dot{\eta} \;+\; g(\eta) \;=\; S\,\tau \;+\; J^{\top}c\;(\eta)\,\lambda \quad (1.1)$$

In questa espressione M(η) ∈ ℝ^{(6+n)×(6+n)} è la matrice di inerzia del sistema — simmetrica, definita positiva, dipendente dalla configurazione; C(η, η̇) ∈ ℝ^{(6+n)×(6+n)} è la matrice dei termini di Coriolis e centrifughi, univocamente determinata richiedendo che Ṁ − 2C sia antisimmetrica; g(η) ∈ ℝ^{(6+n)} è il vettore dei gradienti dell'energia potenziale gravitazionale; S ∈ ℝ^{(6+n)×n} è la matrice di selezione con le prime sei righe nulle; τ ∈ ℝ^n è il vettore delle coppie articolari; J_c(η) ∈ ℝ^{(3k)×(6+n)} è lo Jacobiano dei k punti di contatto e λ ∈ ℝ^{3k} il vettore delle forze di reazione.

La Matrice di Inerzia e il Teorema di König

La struttura della matrice di inerzia per un sistema multi-link si costruisce sommando i contributi di ciascun sottocorpo. Per il corpo i-esimo, avente massa m_i, tensore di inerzia I_i rispetto al proprio centro di massa e Jacobiani J_{v,i}, J_{ω,i} che mappano le velocità generalizzate nelle velocità cartesiane del centro di massa, il contributo alla matrice di inerzia totale è:

$$M_i(\eta) \;=\; m_i J^{\top} v, i\; J_{v,i} \;+\; J^{\top}\omega, i\; R_i I_i R^{\top} i\; J_{\omega,i} \quad (1.2)$$

La matrice totale è M(η) = Σ_{i=1}^{N_b} M_i(η). Il calcolo esplicito di M(η) tramite la formula ricorsiva di Featherstone ha complessità O(n²) nel caso generale, riducibile a O(n) per strutture cinematiche ad albero aperto tramite l'algoritmo CRBA (Composite Rigid Body Algorithm). La proprietà che Ṁ − 2C sia antisimmetrica non è una mera convenienza formale: riflette la conservazione dell'energia meccanica del sistema libero da forze esterne e implica la passività del sistema, proprietà fondamentale per la progettazione di controllori stabili.

1.2 La Matrice Centroidale e la Proiezione dei Momenti

Uno strumento di analisi che ha guadagnato un ruolo centrale nella robotica umanoide moderna è la Centroidal Momentum Matrix (CMM),

introdotta da Orin e Goswami. Definendo il momento centroidale h = (ℓ⊤, k⊤)⊤ ∈ ℝ⁶ dove ℓ = m_tot ṗ_c è il momento lineare totale e k è il momento angolare totale rispetto al centro di massa, la CMM A(η) ∈ ℝ^{6×(6+n)} è la matrice che realizza la proiezione:

$$h = A(\eta)\,\dot{\eta} \quad (1.3)$$

La CMM può essere calcolata ricorsivamente tramite un'estensione dell'algoritmo RNEA in O(n) operazioni. La sua struttura a blocchi A(η) = [A_b(η), A_j(η)] separa il contributo del corpo di base A_b ∈ ℝ^{6×6} da quello dei giunti A_j ∈ ℝ^{6×n}. Una proprietà importante è che A_b coincide, nel frame centroidale, con la matrice di inerzia composta del sistema trattato come corpo rigido, ovvero A_b = [m_tot I_3, 0; 0, I_c] dove I_c è il tensore di inerzia totale rispetto al CoM.

1.3 Il principio di Hamilton e le equazioni di Eulero-Lagrange per il sistema floating-base

Sia Q la varietà di configurazione del sistema, parametrizzata localmente dalle coordinate η = (p_b^⊤, φ_b^⊤, q^⊤)^⊤ ∈ ℝ^{6+n}, dove p_b ∈ ℝ³ denota la posizione del baricentro del torso nel frame mondo, φ_b ∈ ℝ³ è una parametrizzazione locale dell'orientamento del corpo di base (ad esempio gli angoli di Eulero intrinseci ZYX, validi lontano dalle singolarità di rappresentazione) e q ∈ ℝ^n è il vettore delle coordinate articolari. La scelta di una parametrizzazione locale è sufficiente per la derivazione delle equazioni del moto; il trattamento globale su SE(3) × (S¹)^n richiede il formalismo dei gruppi di Lie discusso nella Parte I della monografia precedente.

L'energia cinetica del sistema si scrive come forma quadratica nelle velocità generalizzate:

$$T(\eta, \dot{\eta}) = \tfrac{1}{2}\,\dot{\eta}^{\top} M(\eta)\,\dot{\eta}$$

dove M(η) ∈ ℝ^{(6+n)×(6+n)} è la matrice di inerzia, simmetrica definita positiva, dipendente dalla configurazione. Il potenziale gravitazionale V(η) è una funzione liscia di η, e la funzione di Lagrange è L(η, η̇) = T(η, η̇) − V(η). Il funzionale d'azione sull'intervallo temporale [t_1, t_2] è:

$$S[\eta] = \int_{t_1}^{t_2} L(\eta(t), \dot{\eta}(t))\, dt$$

Il principio di Hamilton afferma che la traiettoria fisica $\eta^*(t)$ rende stazionario il funzionale S tra tutte le curve aventi gli stessi valori agli estremi: $\delta S = 0$ per ogni variazione $\delta\eta$ con $\delta\eta(t_1) = \delta\eta(t_2) = 0$. Calcolando la variazione prima:

$$\delta S = \int_{t_1}^{t_2} \left[\left(\frac{\partial L}{\partial \eta}\right)^{\top} \delta\eta + \left(\frac{\partial L}{\partial \dot{\eta}}\right)^{\top} \delta\dot{\eta} \right] dt$$

Integrando per parti il termine in $\delta\dot{\eta}$ e sfruttando la condizione agli estremi, si ottiene:

$$\delta S = \int_{t_1}^{t_2} \left[\frac{\partial L}{\partial \eta} - \frac{d}{dt}\frac{\partial L}{\partial \dot{\eta}} \right]^{\top} \delta\eta\, dt = 0$$

Il lemma fondamentale del calcolo delle variazioni impone, essendo $\delta\eta$ arbitrario in (t_1, t_2), che l'integrando si annulli identicamente. Si giunge così alle equazioni di Eulero-Lagrange per il sistema non libero:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\eta}} - \frac{\partial L}{\partial \eta} = Q_{est}(\eta, \dot{\eta})$$

dove $Q_{est} \in \mathbb{R}^{6+n}$ è il vettore delle forze generalizzate esterne. Per il principio dei lavori virtuali, un sistema di coppie articolari $\tau \in \mathbb{R}^n$ produce il lavoro virtuale $\delta W_\tau = \tau^{\top} \delta q = (S\tau)^{\top} \delta\eta$, con $S \in \mathbb{R}^{(6+n)\times n}$ matrice di selezione avente le prime sei righe nulle (il corpo di base è privo di attuazione diretta). Le forze di contatto $\lambda \in \mathbb{R}^{3k}$ ai k siti di contatto producono il lavoro virtuale $\delta W_\lambda = \lambda^{\top} J_c(\eta)\, \delta\eta$, quindi $Q_{est} = S\tau + J_c^{\top}(\eta)\, \lambda$.

Il calcolo esplicito di $\frac{d}{dt}\frac{\partial T}{\partial \dot{\eta}} - \frac{\partial T}{\partial \eta}$ richiede di sviluppare separatamente i due addendi. Per la i-esima componente del primo:

$$\frac{\partial T}{\partial \dot{\eta}_i} = \sum_j M_{ij}(\eta)\, \dot{\eta}_j \quad \rightarrow \quad \frac{d}{dt}\frac{\partial T}{\partial \dot{\eta}_i} = \sum_j M_{ij}(\eta)\, \ddot{\eta}_j + \sum_j \dot{M}_{ij}(\eta)\, \dot{\eta}_j$$

La derivata temporale di $M_{ij}(\eta)$ si espande tramite la regola della catena: $\dot{M}_{ij} = \sum_k \frac{\partial M_{ij}}{\partial \eta_k} \dot{\eta}_k$. Per il secondo addendo:

$$\frac{\partial T}{\partial \eta_i} = \tfrac{1}{2} \sum_{j,k} \frac{\partial M_{jk}}{\partial \eta_i}\, \dot{\eta}_j\, \dot{\eta}_k$$

Raccogliendo i termini e scrivendo la doppia somma $\sum_{j,k}$ in modo simmetrico rispetto alle coppie (j,k), si ottiene:

$$[d/dt(\partial T/\partial \dot{\eta}) - \partial T/\partial \eta]_i = \Sigma_j M_{ij}(\eta)\, \ddot{\eta}_j + \Sigma_{j,k} \left[\partial M_{ij}/\partial \eta_k - \tfrac{1}{2}\, \partial M_{jk}/\partial \eta_i\right] \dot{\eta}_j\, \dot{\eta}_k$$

Il secondo addendo ha la struttura di una forma bilineare in $\dot{\eta}$. Introducendo i simboli di Christoffel della seconda specie associati alla metrica riemanniana $g_{ij} \equiv M_{ij}(\eta)$:

$$\Gamma_{ijk}(\eta) = \tfrac{1}{2} \left(\partial M_{ij}/\partial \eta_k + \partial M_{ik}/\partial \eta_j - \partial M_{jk}/\partial \eta_i\right)$$

si definisce la matrice di Coriolis $C(\eta, \dot{\eta})$ con elementi:

$$C_{ij}(\eta, \dot{\eta}) = \Sigma_k \Gamma_{ijk}(\eta)\, \dot{\eta}_k$$

Con questa definizione, il vettore $C(\eta, \dot{\eta})\dot{\eta}$ ha i-esima componente $\Sigma_j C_{ij}\, \dot{\eta}_j = \Sigma_{j,k} \Gamma_{ijk}\, \dot{\eta}_k\, \dot{\eta}_j$, che riproduce esattamente i termini di Coriolis e centrifughi. Si è così ricavato che le equazioni di Eulero-Lagrange prendono la forma canonica:

$$M(\eta)\, \ddot{\eta} + C(\eta, \dot{\eta})\, \dot{\eta} + g(\eta) = S\tau + J_c^{\top}(\eta)\, \lambda$$

dove $g(\eta) = \partial V/\partial \eta$ è il gradiente del potenziale gravitazionale. Questa è precisamente l'equazione (1.1) della monografia, ora derivata rigorosamente dal principio variazionale. Vale la pena sottolineare che la matrice $C(\eta, \dot{\eta})$ definita tramite i simboli di Christoffel non è unica: qualunque altra matrice $\tilde{C}(\eta, \dot{\eta})$ tale che $(\tilde{C} - C)\dot{\eta} = 0$ per ogni $\dot{\eta}$ produce le stesse equazioni del moto. La scelta tramite i simboli di Christoffel è canonica e gode della proprietà di antisimmetria di $\dot{M} - 2C$, che si dimostra nella sezione successiva.

1.4 Simboli di Christoffel, antisimmetria di $\dot{M} - 2C$ e passività

La proprietà che la matrice $\dot{M}(\eta, \dot{\eta}) - 2C(\eta, \dot{\eta})$ sia antisimmetrica non è una convenzione stilistica, ma ha un fondamento geometrico preciso e implicazioni fisiche di primo piano. Geometricamente, essa esprime che la connessione di Levi-Civita della varietà riemanniana (Q, M) è priva di torsione. Fisicamente, essa equivale alla conservazione dell'energia del sistema non soggetto a forze esterne, e implica la proprietà di passività del sistema dinamico.

Proposizione 1.1. *Sia $C(\eta, \dot{\eta})$ la matrice di Coriolis definita dai simboli di Christoffel della metrica $M(\eta)$. Allora la matrice $N(\eta, \dot{\eta}) = \dot{M}(\eta, \dot{\eta}) - 2C(\eta, \dot{\eta})$ è antisimmetrica, ovvero $N^\top = -N$.*

Dimostrazione. Si calcolano gli elementi della matrice $N_{ij} = \dot{M}_{ij} - 2C_{ij}$. Sostituendo le espressioni: $N_{ij} = \sum_k (\partial M_{ij}/\partial \eta_k)\, \dot{\eta}_k - 2 \sum_k \Gamma_{ijk}\, \dot{\eta}_k = \sum_k (\partial M_{ij}/\partial \eta_k)\, \dot{\eta}_k - \sum_k (\partial M_{ij}/\partial \eta_k + \partial M_{ik}/\partial \eta_j - \partial M_{jk}/\partial \eta_i)\, \dot{\eta}_k$. Dopo cancellazione del termine $\sum_k (\partial M_{ij}/\partial \eta_k)\, \dot{\eta}_k$, rimane: $N_{ij} = \sum_k (\partial M_{jk}/\partial \eta_i - \partial M_{ik}/\partial \eta_j)\, \dot{\eta}_k$. Calcolando simmetricamente l'elemento trasposto: $N_{ji} = \sum_k (\partial M_{ik}/\partial \eta_j - \partial M_{jk}/\partial \eta_i)\, \dot{\eta}_k = -N_{ij}$. Poiché questo vale per ogni coppia (i,j) e per ogni $\dot{\eta}$, si conclude $N^\top = -N$. $\square$

La passività del sistema lagrangiano è la conseguenza meccanica più rilevante dell'antisimmetria appena dimostrata. Moltiplicando a sinistra l'equazione del moto per $\dot{\eta}^\top$ in assenza di forze esterne ($\tau = 0$, $\lambda = 0$):

$$\dot{\eta}^\top M(\eta)\, \ddot{\eta} + \dot{\eta}^\top C(\eta, \dot{\eta})\, \dot{\eta} + \dot{\eta}^\top g(\eta) = 0$$

Il termine $\dot{\eta}^\top M(\eta)\, \ddot{\eta} = \mathrm{d}/\mathrm{d}t(\tfrac{1}{2}\, \dot{\eta}^\top M\, \dot{\eta}) - \tfrac{1}{2}\, \dot{\eta}^\top \dot{M}\, \dot{\eta}$. Quindi l'equazione di energia diventa:

$$d/dt(T + V) = \dot{\eta}^\top (S\tau + J_c^\top \lambda) + \tfrac{1}{2}\, \dot{\eta}^\top (\dot{M} - 2C)\, \dot{\eta}$$

Poiché $\dot{M} - 2C$ è antisimmetrica, il quadratic form $\dot{\eta}^\top(\dot{M} - 2C)\, \dot{\eta} = 0$ per ogni $\dot{\eta}$. Pertanto l'equazione di energia si riduce a $\mathrm{d}/\mathrm{d}t(T + V) = \dot{\eta}^\top(S\tau + J_c^\top \lambda)$, che esprime la conservazione dell'energia in assenza di forze esterne e la passività rispetto alla coppia ingresso (τ, λ). Questa proprietà è fondamentale per la sintesi di controllori stabili: un controllore che rispetta la passività non può destabilizzare il sistema, indipendentemente dalle imperfezioni del modello.

La connessione con la geometria riemanniana è la seguente: i termini $C(\eta, \dot{\eta})\dot{\eta}$ rappresentano le forze di Coriolis e centrifughe, che in geometria differenziale corrispondono ai termini di connessione nella derivata covariante. L'equazione del moto libero $M(\eta)\ddot{\eta} + C(\eta, \dot{\eta})\dot{\eta} = 0$ si legge come $\nabla_{\dot{\eta}}\, \dot{\eta} = 0$, ovvero come l'equazione delle geodetiche della varietà

riemanniana (Q, M). I moti liberi del robot (in assenza di gravità e forze esterne) seguono dunque le geodetiche della metrica definita dall'energia cinetica, un risultato che unifica meccanica e geometria differenziale in modo profondo.

1.5 Calcolo ricorsivo della matrice centroidale: l'algoritmo di Orin-Goswami

Il momento centroidale $h = (\ell^\top, k^\top)^\top \in \mathbb{R}^6$, composto dal momento lineare totale $\ell = m_{tot}\, \dot{p}_c$ e dal momento angolare totale k rispetto al centro di massa complessivo, è collegato alle velocità generalizzate dalla Centroidal Momentum Matrix (CMM) $A(\eta) \in \mathbb{R}^{6\times(6+n)}$ secondo la relazione $h = A(\eta)\,\dot{\eta}$. La CMM non è l'intera matrice di inerzia $M(\eta)$, ma le sue prime sei righe nel frame centroidale, dove il frame centroidale ha l'origine nel centro di massa del sistema complessivo e gli assi allineati con il frame mondo.

Il calcolo diretto di $A(\eta)$ tramite differenziazione della cinematica in avanti ha complessità $O(n^2)$ e non è adatto al tempo reale. L'algoritmo di Orin e Goswami (2013) riduce questa complessità a $O(n)$ sfruttando la struttura ricorsiva della cinematica. L'algoritmo si articola in due passate: una passata in avanti che propaga le velocità dal corpo di base alle estremità, e una passata all'indietro che aggrega i contributi al momento centroidale.

Nella passata in avanti si calcolano, per ogni corpo i del sistema (con i che scorre dalla base alle foglie dell'albero cinematico), la velocità del centro di massa del corpo i nel frame centroidale. Sia $v_{c,i} \in \mathbb{R}^3$ la velocità del centro di massa del corpo i e $\omega_i \in \mathbb{R}^3$ la sua velocità angolare. Tramite la formula di composizione delle velocità per corpi rigidi:

$$v_{c,i} = v_{c,parent(i)} + \omega_{parent(i)} \times r_{i,parent} + \omega_i \times r_{c,i}$$

$$\omega_i = \omega_{parent(i)} + R_i\, \hat{z}_i\, \dot{q}_i \quad \textit{(per giunto rotoidale i-esimo)}$$

dove $\hat{z}_i$ è l'asse di rotazione del giunto nel frame locale di corpo i e R_i è la rotazione del frame i rispetto al frame genitore. Queste relazioni possono essere scritte in forma compatta tramite lo Jacobiano della

velocità del centro di massa $J_{v,i}(\eta) \in \mathbb{R}^{3\times(6+n)}$ e dello Jacobiano angolare $J_{\omega,i}(\eta) \in \mathbb{R}^{3\times(6+n)}$, che dipendono solo dalla configurazione η.

Il contributo del corpo i al momento centroidale è:

$$h_i = [m_i\, v_{c,i};\ I_i^c\, \omega_i + m_i\, (p_{c,i} - p_c) \times v_{c,i}]$$

dove I_i^c è il tensore di inerzia del corpo i rispetto al proprio centro di massa espresso nel frame centroidale, e $p_{c,i} - p_c$ è il vettore dal centro di massa del sistema al centro di massa del corpo i. Esprimendo h_i in termini di $\dot{\eta}$ tramite gli Jacobiani si ottiene $h_i = A_i(\eta)\,\dot{\eta}$, con:

$$A_i(\eta) = [m_i\, J_{v,i}(\eta);\ I_i^c(\eta)\, J_{\omega,i}(\eta) + m_i\, [p_{c,i} - p_c]^\times\, J_{v,i}(\eta)]$$

dove $[\cdot]^\times$ denota la hat map. La CMM totale è $A(\eta) = \Sigma_i\, A_i(\eta)$. Il calcolo ricorsivo di questo insieme di Jacobiani richiede $O(n)$ operazioni grazie alla struttura triangolare inferiore della cinematica in avanti: $J_{v,i}$ e $J_{\omega,i}$ differiscono da $J_{v,\text{parent}(i)}$ e $J_{\omega,\text{parent}(i)}$ solo per le colonne relative ai giunti sulla catena cinematica che collega la base al corpo i.

La struttura a blocchi $A(\eta) = [A_b(\eta), A_j(\eta)]$ separa il contributo del corpo di base $A_b \in \mathbb{R}^{6\times6}$ da quello articolare $A_j \in \mathbb{R}^{6\times n}$. Una proprietà notevole è che A_b, calcolata nel frame centroidale, coincide con la matrice di inerzia del sistema trattato come corpo rigido equivalente: $A_b = \text{diag}(m_{tot}\, I_3,\ I_c)$, dove $I_c \in \mathbb{R}^{3\times3}$ è il tensore di inerzia totale rispetto al centro di massa del sistema, simmetrico definito positivo. Questa struttura consente di scrivere le equazioni del moto centroidale in una forma particolarmente trasparente per il controllo: $\dot{h} = A(\eta)\,\ddot{\eta} + \dot{A}(\eta, \dot{\eta})\,\dot{\eta} = \Sigma_i\, (f_i;\ p_{c,i} \times f_i + n_i) + (m_{tot}\, g;\ 0)$, che sarà il punto di partenza del controllore MPC nel capitolo successivo.

1.6 Modelli di Attrito Avanzati per Giunti Robotici: Stribeck e LuGre

I modelli di attrito classici adottati nelle equazioni del moto (§1.1) assumono una caratteristica viscosa lineare $\tau_{fric} = B\,\dot{q}$, che è sufficiente per il calcolo numerico stabile ma non cattura i fenomeni di attrito che

dominano il comportamento dei giunti reali nelle fasi di transizione da stazionario a moto. Queste transizioni — denominate presliding regime — producono effetti di stick-slip che causano micro-oscillazioni indesiderate durante i task manipolativi di precisione e compromettono la stima dei parametri inerziali tramite system identification.

Il modello di Stribeck

Il modello di Stribeck (1902), riscoperto nell'ambito della tribologia moderna, descrive la forza di attrito come funzione non lineare della velocità relativa delle superfici in contatto. La caratteristica di Stribeck $\tau\_S: \mathbb{R} \to \mathbb{R}$ è:

$$\tau\_S(\dot{q}) = [\tau\_c + (\tau\_s - \tau\_c) \exp(-(\dot{q}/v\_S)^2)] \operatorname{sign}(\dot{q}) + \sigma\_v \dot{q}$$

dove $\tau\_c > 0$ è la coppia di attrito dinamico (Coulomb), $\tau\_s > \tau\_c$ è la coppia di attrito statico (breakaway torque), v_S è la velocità di Stribeck che separa i regimi di lubrificazione e σ_v è il coefficiente di viscosità. Il termine esponenziale cattura il picco di attrito alla transizione dallo stazionario al moto e il successivo decremento (curva di Stribeck propriamente detta) dovuto alla formazione dello strato di lubrificante idrodinamico. Per $v\_S \to 0$, il modello di Stribeck collassa al modello Coulomb-viscoso standard.

L'identificazione dei parametri (τ_c, τ_s, v_S, σ_v) si effettua tramite prove a velocità imposta $\dot{q} = \text{cost}$, misurando la coppia di regime per diversi valori di $\dot{q}$ e adattando la curva con regressione non lineare ai minimi quadrati (§21.1). La complessità di questo approccio risiede nella necessità di velocità molto basse ($\dot{q} \approx 0.001$–0.01 rad/s) per eccitare la transizione di Stribeck, che richiede driver di corrente a basso rumore e encoder ad alta risoluzione.

Il modello LuGre

Il modello LuGre (Canudas de Wit et al., 1995) è il modello dinamico di attrito più adottato nella letteratura di controllo per robotica. A differenza dei modelli statici come Stribeck, LuGre incorpora una variabile interna $z \in \mathbb{R}$ che rappresenta la deformazione media delle asperità del contatto (bristle displacement). Le equazioni del modello sono:

$$\dot{z} = \dot{q} - \frac{\sigma_0 |\dot{q}|}{g(\dot{q})} z$$

$$\tau_{fric} = \sigma_0 z + \sigma_1 \dot{z} + \sigma_v \dot{q}$$

$$g(\dot{q}) = \tau_c + (\tau_s - \tau_c) \exp(-(\dot{q}/v_S)^2)$$

dove $\sigma_0 > 0$ è la rigidezza delle asperità, $\sigma_1 \geq 0$ è lo smorzamento delle asperità, e $g(\dot{q})$ è la funzione di Stribeck che scala la deformazione in regime stazionario. La variabile z ha dimensioni fisiche di spostamento lineare (mm) ed è una misura della deformazione elastica microscopica delle asperità nel contatto. In regime stazionario ($\dot{z} = 0$), il modello si riduce a $\tau_{fric} = \sigma_0 z_\infty + \sigma_v \dot{q}$ con $z_\infty = g(\dot{q})\, \mathrm{sign}(\dot{q}) / \sigma_0$, riproducendo la caratteristica di Stribeck.

Proposizione 1.2 (Passività del modello LuGre).

Il modello LuGre con $\sigma_1 > 0$ è un sistema passivo rispetto alla coppia di attrito τ_{fric} e alla velocità $\dot{q}$: la funzione $V_L = \frac{1}{2} \sigma_0 z^2 \geq 0$ soddisfa $\dot{V}_L = -\sigma_1 \dot{z}^2 - \sigma_v \dot{q}^2 + \tau_{fric} \dot{q}$, ovvero $dV_L/dt \leq \tau_{fric} \dot{q}$, il che implica che il sistema non genera energia dal nulla.

Dimostrazione. La derivata di V_L lungo le traiettorie del modello è $\dot{V}_L = \sigma_0 z \dot{z} = \sigma_0 z [\dot{q} - (\sigma_0 |\dot{q}|/g) z] = \sigma_0 z \dot{q} - \sigma_0^2 |\dot{q}| z^2/g$. Sostituendo nell'espressione della potenza dissipata $\tau_{fric} \dot{q} = (\sigma_0 z + \sigma_1 \dot{z} + \sigma_v \dot{q}) \dot{q}$ e riarrangiando si ottiene la disuguaglianza di passività. □

L'integrazione del modello LuGre nelle equazioni del moto lagrangiane (1.1) sostituisce il termine $B\dot{q}$ con $\tau_{fric}(\dot{q}, z)$ e richiede l'integrazione numerica congiunta dello stato del robot $(\eta, \dot{\eta})$ e della variabile di attrito $z \in \mathbb{R}^n$ (uno per giunto). Il sistema risultante è un DAE aumentato di dimensione $2(6+n)+n$, che rimane comunque trattabile con i solver ODE disponibili in robot.

Capitolo 2 — Whole-Body Control con MPC su SE(3)

Il controllo dell'intero corpo di un umanoide — comunemente indicato con l'acronimo WBC, Whole-Body Control — è il problema di calcolare, ad ogni istante di tempo, le coppie articolari τ e le forze di contatto λ che consentono al sistema di eseguire simultaneamente un insieme di obiettivi cinematici e dinamici, rispettando i vincoli fisici imposti dall'architettura meccanica e dalle leggi di attrito. La difficoltà fondamentale risiede nella ridondanza del sistema: un umanoide con 30 o più gradi di libertà e soltanto sei vincoli di contatto possiede un'abbondanza di soluzioni cinematiche, tra le quali occorre selezionare quella ottimale secondo un criterio razionale.

2.1 Dinamica Centroidale e Pianificazione MPC

Le equazioni centroidali descrivono l'evoluzione di ℓ e k in funzione delle forze e dei momenti ai punti di contatto. Raccogliendo i momenti nel vettore h = (ℓ⊤, k⊤)⊤, la dinamica ha la forma:

$$\dot{h} = \sum_{i=1}^{k} \left(f_i ;\ p_{ci} \times f_i + n_i \right) + \left(mtot\ g ;\ 0 \right) \quad (2.1)$$

dove f_i ∈ ℝ³ e n_i ∈ ℝ³ sono rispettivamente la forza e il momento risultanti al sito di contatto i-esimo, e p_{ci} è il vettore dal CoM al punto di contatto i. Si definisce lo stato centroidale aumentato x_k = (p_{c,k}⊤, ℓ_k⊤, k_k⊤)⊤ ∈ ℝ⁹ e il vettore di controllo u_k = (f_{1,k}⊤, …, f_{K,k}⊤)⊤ ∈ ℝ^{3K}. Le dinamiche discrete linearizzate attorno al CoM sono:

$$x_{k+1} = A_d x_k + B_d(p_k) u_k + d_k \quad (2.2)$$

dove A_d e B_d(p_k) sono le matrici di sistema e di controllo risultanti dalla discretizzazione Eulero esplicita della dinamica centroidale lineare, e d_k raccoglie il termine gravitazionale. La matrice B_d dipende dalle posizioni dei piedi p_k e quindi varia ad ogni passo, rendendo il sistema linear-parametricamente variante (LPV). Il problema MPC su un orizzonte di N passi temporali è il programma quadratico vincolato:

$$\min_{\{u_k\}} \sum_{k=0}^{N-1} \left[\| x^{ref}_{k+1} - x_{k+1} \|^2_Q + \| u_k \|^2_R \right] \quad (2.3)$$

soggetto ai vincoli dinamici (2.2), ai vincoli di contatto unilaterale f_{z,i,k} ≥ 0, ai limiti di forza tangenziale secondo il cono di attrito ‖f_{t,i,k}‖ ≤ μ f_{z,i,k} (approssimato come cono lineare per mantenere la struttura QP), e ai limiti sullo spostamento del CoP dentro il poligono di supporto.

2.2 Whole-Body Control come QP Gerarchico

Dato che il MPC centroidale produce una traiettoria di riferimento per i momenti e le forze di contatto ottimali, il livello inferiore del WBC deve trovare configurazione articolare, velocità e coppie che realizzino tale traiettoria rispettando tutti i vincoli del corpo intero. Questo problema è formulato come una sequenza di programmi quadratici ordinati per priorità (Hierarchical QP, HQP). Il WBC al k-esimo livello di priorità risolve:

$$min_{\ \delta\ddot{\eta},\,\tau,\,\lambda}\ \|J_k(\eta)\,\delta\ddot{\eta} - \dot{x}_k\|^2 + \varepsilon\,\|\delta\ddot{\eta}\|^2 \quad (2.4)$$

soggetto a: M(η)δη̈ + h(η, η̇) = Sτ + J_c⊤λ (dinamiche del corpo intero), J_c(η)δη̈ + J̇_c η̇ = 0 (vincoli cinematici di contatto costante), e le soluzioni dei livelli superiori proiettate nel null-space dello Jacobiano J_{k−1} tramite il proiettore P_k = I − J†_{k−1} J_{k−1}.

> **Teorema 2.1 (Consistenza della gerarchia WBC).**
>
> *Sia {J₁, J₂, …, J_m} un insieme ordinato di task Jacobiani con J_k ∈ ℝ^{r_k×n}. L'HQP gerarchico con proiezione nel null-space sequenziale produce, ad ogni passo, coppie τ* ∈ ℝ^n che sono soluzione esatta del task di priorità k se e solo se il rango del Jacobiano aumentato [J₁⊤, …, J_k⊤]⊤ non eccede n. In caso contrario, la soluzione è il minimo norma nell'insieme delle soluzioni fattibili per i task di priorità superiore.*

Dimostrazione. Segue dalla costruzione per induzione del proiettore composito P_k, che è la proiezione ortogonale sul null-space di [J₁⊤, …, J_{k-1}⊤]⊤. Poiché ogni J_k è applicato nella forma J_k P_k, l'effettivo Jacobiano del k-esimo task nel sottospazio residuo ha rango min(r_k, dim ker P_{k-1}). Il risultato segue dal teorema di fattorizzazione SVD e dalla monotonicità del null-space sotto composizione. □

2.3 Errore Geometrico su SE(3) e Controllo Intrinseco

La formulazione su SE(3) è necessaria quando si vuole mantenere la correttezza geometrica nella traiettoria dell'organo terminale e del torso. Data una task di tracking dell'orientamento, l'errore logaritmico nel gruppo SO(3) è:

$$e_R = Log\left(R^{d\top} R\right) \in so(3) \cong \mathbb{R}^3 \quad (2.5)$$

e la sua derivata temporale è ė_R = J_R(q) q̇, dove J_R è lo Jacobiano differenziale dell'errore di orientazione. Per un task combinato di posizione e orientazione in SE(3), l'errore nel frame dello spazio tangente è e_{SE(3)} = Log_{SE(3)}(T^{d⊤} · T) ∈ se(3) ≅ ℝ⁶, dove Log_{SE(3)}: SE(3) → se(3) è il logaritmo di gruppo definito dalla formula di Rodrigues estesa. Il controllore task-space corrispondente è:

$$\dot{x}_d = \dot{x}_{ref} + K_p e_{SE(3)} + K_d \dot{e}_{SE(3)} \quad (2.6)$$

Questo controllore è detto intrinseco perché opera direttamente sull'algebra di Lie se(3) senza richiedere la parametrizzazione esplicita dell'orientamento tramite angoli di Eulero o quaternioni, evitando le singolarità di rappresentazione che affliggono i formalismi estrinseci.

2.4 Stabilità del WBC: passività, funzioni di Lyapunov e analisi dei guadagni

La stabilità del WBC gerarchico è un problema non banale per almeno due ragioni: la natura dinamicamente inconsistente del sistema (il numero di gradi di libertà attuati è inferiore al numero di vincoli se il sistema è ridondante), e il fatto che il QP gerarchico non produce, in generale, una legge di controllo espressa in forma chiusa, rendendo inapplicabili i classici teoremi di Lyapunov per sistemi a feedback stazionario.

La via più produttiva alla stabilità è quella dell'analisi energetica basata sulla passività. Si considera il task-space controller di più alta priorità, il cui obiettivo è portare un punto di controllo cartesiano x ∈ ℝ^m al riferimento x_d(t). L'errore di tracking è e(t) = x_d(t) − x(η(t)) ∈ ℝ^m, e la

velocità di errore è $\dot{e}(t) = \dot{x}_d(t) - J_1(\eta)\,\dot{\eta}(t)$, dove $J_1(\eta) \in \mathbb{R}^{m\times(6+n)}$ è lo Jacobiano del task.

Invertendo cinematicamente e applicando il controllore task-space PD nel null-space della dinamica del corpo intero, la legge di controllo risultante al livello di coppia è, nella sua forma più semplice priva di compensazione della dinamica:

$$\tau_1 = J_1^{\dagger}(\eta)\,[K_p\, e + K_d\, \dot{e}] + [I - J_1^{\dagger}(\eta)\, J_1(\eta)]\,\tau_0$$

dove $J_1^{\dagger}(\eta) = M^{-1} J_1^{\top} (J_1 M^{-1} J_1^{\top})^{-1}$ è lo pseudoinverso pesato dalla metrica M e τ_0 è la coppia di grado inferiore di priorità. Con questa scelta, la chiusura del loop nel sottospazio del task primario produce l'errore di task-dynamics:

$$M_{task}(\eta)\,\ddot{e} + K_d\,\dot{e} + K_p\, e = \varepsilon(\eta, \dot{\eta})$$

dove $M_{task}(\eta) = (J_1(\eta)\, M^{-1}(\eta)\, J_1^{\top}(\eta))^{-1} \in \mathbb{R}^{m\times m}$ è la matrice di inerzia operazionale, e $\varepsilon(\eta, \dot{\eta})$ raccoglie i termini di disaccoppiamento residui (termini di Coriolis e centrifughi non compensati, che sono $O(\|\dot{\eta}\|^2)$). Nell'approssimazione lineare, che è valida nella regione in cui $|\dot{\eta}|$ è piccola rispetto al raggio di curvatura della varietà, $\varepsilon \approx 0$ e l'equazione dell'errore è:

$$M_{task}(\eta)\,\ddot{e} + K_d\,\dot{e} + K_p\, e \approx 0$$

Questa è un'equazione di un oscillatore smorzato parametrizzato. Si propone la funzione di Lyapunov candidata:

$$V_1(e, \dot{e}) = \tfrac{1}{2}\,\dot{e}^{\top} M_{task}(\eta)\,\dot{e} + \tfrac{1}{2}\, e^{\top} K_p\, e + \alpha\, e^{\top} M_{task}(\eta)\,\dot{e}$$

dove $\alpha > 0$ è un parametro scalare. V_1 è definita positiva se e solo se:

$$K_p - \alpha^2\, M_{task}(\eta) \succ 0, \quad \text{ossia} \quad \lambda_{min}(K_p) > \alpha^2\, \|M_{task}\|_2$$

La derivata temporale di V_1 lungo le traiettorie del sistema è:

$$\dot{V}_1 = \dot{e}^{\top} M_{task}\,\ddot{e} + \tfrac{1}{2}\,\dot{e}^{\top} \dot{M}_{task}\,\dot{e} + e^{\top} K_p\,\dot{e} + \alpha\,\dot{e}^{\top} M_{task}\,\dot{e} + \alpha\, e^{\top} \dot{M}_{task}\,\dot{e} + \alpha\, e^{\top} M_{task}\,\ddot{e}$$

Sostituendo l'espressione approssimata $M_{task}\,\ddot{e} \approx -K_d\,\dot{e} - K_p\, e$ e raccogliendo:

$$\dot{V}_1 = -\dot{e}^\top K_d \dot{e} - \alpha\, e^\top K_p\, e + \alpha\, \dot{e}^\top (\tfrac{1}{2}\dot{M}_{task} - K_d)\,\dot{e} + (\tfrac{1}{2}\,\dot{e}^\top \dot{M}_{task}\,\dot{e} + \alpha\, e^\top \dot{M}_{task}\,\dot{e})$$

Il termine $\tfrac{1}{2}\,\dot{e}^\top \dot{M}_{task}\,\dot{e}$ è bounded da $\|\dot{M}_{task}\|_2\,\|\dot{e}\|^2/2$. Scegliendo $K_d > (\tfrac{1}{2}\,\|\dot{M}_{task}\|_2 + \beta)\,I$ per qualche $\beta > 0$ e α sufficientemente piccolo, si può mostrare che $\dot{V}_1 \le -\beta(\|\dot{e}\|^2 + \|e\|^2) < 0$ per $(e, \dot{e}) \neq 0$ in un intorno dell'equilibrio, garantendo la stabilità asintotica locale dell'errore di tracking del task primario. La regione di attrazione è determinata dal livello massimo di V_1 all'interno del quale le stime lineari rimangono valide.

> **Teorema 2.2 (Stabilità locale del task-space controller).** *Siano $K_p, K_d \in \mathbb{R}^{m\times m}$ matrici simmetriche definite positive con $\lambda_{min}(K_p) > \alpha^2 \sup_\eta \|M_{task}(\eta)\|_2$ e $K_d > \tfrac{1}{2} \sup_{\eta,\dot{\eta}:\, \|\eta\|\le\rho} \|\dot{M}_{task}(\eta,\dot{\eta})\|_2 \cdot I$ per qualche $\rho > 0$. Allora il controllore PD intrinseco in task-space è asintoticamente stabile per condizioni iniziali in un intorno dell'equilibrio di raggio $O(\rho)$ nelle norme $(\|e\|, \|\dot{e}\|)$.*
>
> *Dimostrazione.* Si adotta la funzione di Lyapunov V_1 sopra definita. Si verifica che V_1 è definita positiva sotto le condizioni sui guadagni, e che $\dot{V}_1 \le -\beta(\|e\|^2 + \|\dot{e}\|^2)$ nella regione $\{\|\eta\| \le \varrho\}$ dove le stime lineari sono valide. Il Teorema di Lyapunov sulla stabilità asintotica locale garantisce che ogni traiettoria con $(e(0), \dot{e}(0))$ in un sottolivello di V_1 converge all'origine. □

Un aspetto più sottile riguarda l'interazione tra i livelli gerarchici. L'analisi di stabilità per il task secondario (null-space) è più complessa: la proiezione nel null-space di J_1 introduce un accoppiamento indiretto tra i task che modifica la matrice di inerzia operazionale effettiva. In generale, la stabilità globale della gerarchia non può essere dedotta dalla stabilità locale di ciascun livello, e richiede un'analisi di Lyapunov composta, o l'ipotesi di separazione temporale delle scale dei guadagni.

2.5 Il QP gerarchico in tempo reale: struttura algebrica e algoritmi efficienti

La risoluzione del QP gerarchico in tempo reale — tipicamente con frequenza di 500 Hz – 1 kHz — è il collo di bottiglia computazionale del WBC. Ad ogni istante di campionamento si deve risolvere una sequenza di programmi quadratici della forma:

$$\min_{x} \tfrac{1}{2} \|A_k x - b_k\|^2 \ \text{s.t.} \ A_i x = b_i,\ i = 1, \ldots, k-1 \ \text{e} \ C x \geq d$$

dove $x = (\delta\ddot{\eta}^\top, \tau^\top, \lambda^\top)^\top \in \mathbb{R}^{n_x}$ è il vettore delle variabili di decisione, composto dall'accelerazione generalizzata, dalle coppie articolari e dalle forze di contatto. La dimensionalità tipica è $n_x = 6 + n + n + 3k \approx 100–200$ per un umanoide a 30 gradi di libertà con 4 siti di contatto.

La struttura del QP consente di sfruttare il complemento di Schur per eliminare le variabili di aggiornamento dinamico. Partizionando il sistema vincolato nelle variabili (z_1, z_2) dove $z_1 \in \mathbb{R}^{n_c}$ sono le variabili vincolate dall'uguaglianza e $z_2 \in \mathbb{R}^{n_x - n_c}$ sono le rimanenti, si ha, dopo la proiezione nel null-space dei vincoli di uguaglianza tramite Z (base del ker(A)):

$$x = x_p + Z y, \ \text{dove} \ A x_p = b \ \text{(soluzione particolare)}, \ A Z = 0$$

Il QP ridotto nelle variabili libere $y \in \mathbb{R}^{n_x - n_c}$ è:

$$\min_y \tfrac{1}{2} \|A_k (x_p + Zy) - b_k\|^2 = \min_y \tfrac{1}{2} \|(A_k Z) y - (b_k - A_k x_p)\|^2$$

che è un problema di minimi quadrati non vincolato, risolubile in forma chiusa tramite la decomposizione QR della matrice $A_k Z$. La complessità complessiva di questa riduzione è $O(n_x^2 \cdot n_c)$ per la fattorizzazione del null-space, e $O(n_x^2)$ per ogni task aggiuntivo, scalando favorevolmente rispetto ai $O(n_x^3)$ di un solutore QP generico.

Per la componente di disuguaglianza — i vincoli di attrito e unilateralità — si adotta il metodo del set attivo (active-set method). All'i-esima iterazione si mantiene un insieme attivo $A_i \subseteq \{1, \ldots, n_{ineq}\}$ di vincoli trattati come uguaglianze, si risolve il QP di uguaglianza corrispondente e si verificano le condizioni di ottimalità di Karush-Kuhn-Tucker (KKT). Se un moltiplicatore di Lagrange è negativo per un vincolo nell'insieme attivo, quel vincolo viene rimosso; se un vincolo violato esiste nell'insieme inattivo, viene aggiunto. La convergenza è garantita in un numero finito

di passi (al più 2^{n_{ineq}} per QP non degeneri), ma in pratica il warm-starting — inizializzazione con l'insieme attivo dell'istante precedente — riduce il numero di iterazioni a 1-3 per ciclo di controllo, consentendo l'esecuzione in tempo reale.

Un ulteriore guadagno computazionale si ottiene sfruttando la struttura sparsa della matrice di inerzia M(η). Per sistemi a cinematica ad albero aperto, M(η) è densa in generale; tuttavia, nella struttura di Featherstone, la matrice di inerzia composta (CRBA) può essere calcolata in $O(n^2)$ e il risultato può essere factorizzato con la decomposizione di Cholesky in $O(n^3)$. Per gli umanoidi moderni con n ≈ 30, il calcolo di M a 1 kHz richiede circa 10–50 μs su un processore ARM moderno, un onere assolutamente compatibile con il ciclo di controllo.

2.5 Complessità Computazionale del WBC: RNEA, CRBA e Solver QP

La realizzazione del WBC in tempo reale su un umanoide con n gradi di libertà richiede, ad ogni ciclo di controllo a 1 kHz, il completamento di tre passi computazionali: (1) il calcolo della dinamica del sistema tramite algoritmi ricorsivi, (2) la costruzione e risoluzione del QP gerarchico, e (3) la conversione delle coppie ottimali in segnali di riferimento per gli attuatori. La finestra temporale disponibile per l'intero ciclo è di 1 ms, di cui circa il 60–70% è consumato dai passi (1) e (2).

Algoritmo RNEA e Complessità O(n)

Il calcolo del vettore dei termini non lineari $h\left(\eta, \dot{\eta}\right) = C\left(\eta, \dot{\eta}\right)\dot{\eta} + g(\eta)$ è il passo computazionale più frequente nel WBC, in quanto viene eseguito ad ogni valutazione del gradiente del QP. L'**algoritmo RNEA** (Recursive Newton-Euler Algorithm, Luh-Walker-Paul 1980) calcola $h\left(\eta, \dot{\eta}\right)$ in $O(n)$ operazioni tramite due passate ricorsive sulla struttura ad albero della cinematica.

Passata in avanti (dalla radice alle foglie): per ogni corpo i in ordine topologico, si propagano le velocità e accelerazioni:

$$\dot{V}_i = {}^{i}X_{i-1}\dot{V}_{i-1} + S_i\ddot{q}_i + \dot{S}_i\dot{q}_i$$

$$\ddot{V}_i = {}^{i}X_{i-1}\ddot{V}_{i-1} + S_i\ddot{q}_i + \dot{S}_i\dot{q}_i$$

dove ${}^{i}X_{i-1} \in R^{6\times6}$ è la matrice di trasformazione spaziale dal frame $i-1$ al frame i (in notazione di Featherstone), $S_i \in R^6$ è il vettore di moto del giunto i (per un giunto rotoidale, $S_i = (0,0,1,0,0,0)^\top$ nello spazio delle torsioni), e $\dot{V}_i = \left(\omega_i, v_{c,i}\right)^\top \in R^6$ è la velocità spaziale del corpo i.

Passata all'indietro (dalle foglie alla radice): per ogni corpo i in ordine inverso, si calcolano le forze spaziali:

$$F_i = I_i^c\ddot{V}_i + V_i\times^* I_i^c V_i - {}^{i}X_0 f_i^{ext}$$

$$\tau_i = S_i^\top F_i$$

dove $I_i^c \in R^{6\times6}$ è la matrice di inerzia spaziale del corpo i, $V_i\times^*$ è il prodotto incrociato spaziale aggiunto, e F_i accumula la forza spaziale trasmessa al corpo genitore. Ogni operazione è $O(1)$ per strutture cinematiche ad albero aperto, e la complessità totale è $O(n)$ con costante moltiplicativa di circa $60n$ operazioni floating-point per n gradi di libertà.

Teorema 2.2 (Ottimalità di RNEA). *Non esiste un algoritmo che calcoli il vettore* $h\left(\eta,\dot{\eta}\right)$ *per un sistema multi-corpo ad albero aperto generico con meno di* $\Omega(n)$ *operazioni floating-point.*

Dimostrazione. Il lower bound segue dalla struttura del problema: ogni giunto contribuisce almeno con un termine di Coriolis della forma $C_{ij}\dot{q}_i\dot{q}_j$ che dipende da almeno una operazione non banale. Poiché esistono n giunti e il contributo di ciascuno non è calcolabile senza accedere ai dati cinematici del rispettivo corpo, la complessità è $\Omega(n)$. L'algoritmo RNEA raggiunge questo bound. ▫

Algoritmo CRBA e Struttura Sparse della Matrice di Inerzia

Il calcolo esplicito della matrice di inerzia $M(\eta) \in R^{(6+n)\times(6+n)}$ è necessario per la costruzione del QP gerarchico. L'algoritmo **CRBA** (Composite Rigid

Body Algorithm, Walker-Orin 1982) lo calcola in $O\left(n^2\right)$ operazioni sfruttando la ricorsione sull'albero cinematico.

Per una struttura cinematica ad albero con profondità media d, la matrice $M(\eta)$ ha una struttura di **sparsità gerarchica**: l'elemento M_{ij} è non nullo se e solo se i giunti i e j appartengono allo stesso cammino dalla radice a una foglia. Per un robot umanoide con struttura cinematica tipica (torso, due braccia, due gambe), la matrice M ha $O(n)$ elementi non nulli per strutture ad albero bilanciato di profondità $O(logn)$.

L'algoritmo CRBA per il calcolo di $M(\eta)$ opera come segue. Si definisce la **matrice di inerzia composita** $\hat{I}_i(\eta)$ del sottosistema radicato in i come la somma delle inerzie spaziali di tutti i corpi nel sottoalbero:

$$\hat{I}_i(\eta) = I_i^c + \sum_{j \in subtree(i)} {}^iX_j I_j^c \left({}^iX_j\right)^\top$$

La colonna j della matrice di inerzia è allora $M_{:,j} = {}^iX_j^\top \hat{I}_i S_j$ per ogni giunto j con corpo i come antenato. Il costo totale è $O\left(n^2\right)$ nel caso generale, riducibile a $O(nd)$ per strutture ad albero con profondità $d \ll n$.

Struttura Sparse del QP Gerarchico e Solver

Il QP gerarchico del WBC ha una struttura che può essere sfruttata per ridurre il costo computazionale rispetto a un QP denso. Il problema di livello k è:

$$\min_{\ddot{\delta\eta},\tau,\lambda} \| J_k(\eta)\ddot{\delta\eta} - \dot{x}_k\|^2 + \varepsilon\|\ddot{\delta\eta}\|^2$$

soggetto a:

$$M(\eta)\ddot{\delta\eta} + h\left(\eta, \dot{\eta}\right) = S\tau + J_c^\top \lambda$$

$$J_c(\eta)\ddot{\delta\eta} + \dot{J}_c\dot{\eta} = 0$$

$$\tau_{min} \leq \tau \leq \tau_{max}, \quad \|f_{t,i}\| \leq \mu f_{z,i}$$

Le matrici di vincolo hanno struttura sparse: $M(\eta)$ ha la sparsità gerarchica descritta sopra, $J_c(\eta) \in R^{3k\times(6+n)}$ ha $O(kn)$ elementi non nulli (al più n per riga), e la matrice dei coni di attrito ha struttura diagonale a blocchi. Questa sparsità riduce il costo di fattorizzazione del sistema KKT da $O\left((6+n+3k)^3\right)$ del caso denso a $O\left((6+n+3k)n^2\right)$ in media.

I solver più adottati nella pratica del WBC in tempo reale sono:

OSQP (Operator Splitting Quadratic Program, Stellato et al. 2020): risolve il QP tramite l'algoritmo ADMM (Alternating Direction Method of Multipliers), con convergenza garantita in un numero fisso di iterazioni e tolleranza programmabile. La complessità per iterazione è $O\left(n^{1.5}\right)$ per matrici sparse, e la convergenza a precisione dell'ordine 10^{-4} richiede tipicamente 50–200 iterazioni per problemi di dimensione $n\leq 50$. Il vantaggio principale di OSQP è la sua robustezza a problemi mal condizionati: l'algoritmo ADMM converge anche quando la matrice Hessiana non è definita positiva in senso stretto.

qpOASES (Ferreau et al. 2014): implementa il metodo del simplesso attivo per QP densi e semi-densi. Per il WBC, la warm-starting — l'inizializzazione del solver alla soluzione del ciclo precedente — riduce il numero di passi dell'algoritmo del simplesso a 1–5 per perturbazioni piccole del problema, ottenendo tempi di soluzione sub-millisecondo per $n\leq 60$.

Proximal DDP (Differential Dynamic Programming prossimale): per formulazioni MPC del WBC con orizzonte corto ($N\leq 5$ passi), il DDP con penalizzazione prossimale sfrutta la struttura a banda della matrice di Riccati e raggiunge complessità $O\left(Nn^3\right)$ per ciclo, con N piccolo. Questo approccio è adottato in Crocoddyl (Mastalli et al. 2022), il framework open-source di WBC più diffuso nella ricerca.

Proposizione 2.2 (Budget computazionale del WBC a 1 kHz). *Sia n il numero di gradi di libertà del robot e k il numero di contatti attivi. Il ciclo WBC a 1 kHz è realizzabile con hardware commodity (≤ 10 GFLOPS) se e solo se $n^2 + kn \leq C$ per una costante $C \approx 5\times 10^3$ che dipende dall'implementazione. Per $n = 30$ e $k = 4$, il budget è rispettato con ampio margine.*

Dimostrazione. RNEA richiede $60n$ FLOP, CRBA richiede $O\left(n^2\right)$ FLOP con costante ≈ 10, la costruzione del QP richiede $O\left(kn^2\right)$ FLOP, e OSQP con 100 iterazioni richiede $O\left(100 \cdot n^{1.5}\right)$ FLOP. Per $n = 30, k = 4$: RNEA ≈ 1800, CRBA ≈ 9000, costruzione QP ≈ 36000, solver ≈ 16000 FLOP totali ≈ 63000 per ciclo. A 10 GFLOPS disponibili, il ciclo a 1 kHz richiede 6.3×10^4 FLOP, ampiamente al di sotto del budget di 10^7 FLOP per ciclo da 1 ms. ▫

2.6 Controllo Adattivo e Stima Parametrica Online nel WBC

Il controllore WBC del Capitolo 2 presuppone la conoscenza accurata della matrice di inerzia M(η) e del vettore dei termini non lineari h(η, η̇). Nella pratica, i parametri inerziali stimati tramite system identification (§21.1) presentano errori del 5–15% dovuti a carichi utili variabili, deformazioni elastiche dei link e usura dei giunti. Il controllo adattivo integra la stima dei parametri nel loop di controllo, aggiornando la conoscenza del modello in tempo reale.

Il controllore adattivo di Slotine-Li

Il controllore adattivo di Slotine e Li (1987), originariamente derivato per manipolatori seriali fissi, si generalizza ai sistemi floating-base con la stessa struttura formale. Si definisce la velocità di riferimento η̇_r = η̇_d − Λ ẽ dove ẽ = η − η_d è l'errore di tracking, Λ = diag(λ₁,...,λ_{6+n}) > 0 e η_d è il riferimento desiderato. Il segnale di errore di sliding s = η̇ − η̇_r = ė + Λ ẽ racchiude sia l'errore di posizione che di velocità.

La legge di controllo adattiva è:

$$\tau = Y(\eta, \dot{\eta}, \dot{\eta}_r, \ddot{\eta}_r)\,\hat{\theta} - K_D\, s$$

dove $Y \in \mathbb{R}^{n \times p}$ è il regressore del robot (come in §21.1), $\hat{\theta} \in \mathbb{R}^p$ è la stima corrente dei parametri inerziali e $K_D > 0$ è la matrice di smorzamento. La legge di aggiornamento dei parametri è:

$$d\hat{\theta}/dt = -\Gamma\, Y^T\, s$$

con $\Gamma = \Gamma^T > 0$ matrice di guadagno di adattamento. La scelta di questa legge di aggiornamento — che è il gradiente discendente della funzione di Lyapunov rispetto all'errore di parametri — garantisce la convergenza dell'errore $s \to 0$.

Teorema 2.3 (Stabilità globale del controllore adattivo Slotine-Li).

Sotto l'ipotesi che i parametri reali θ siano costanti e che il sistema sia persistentemente eccitato, il controllore adattivo con legge di aggiornamento $d\hat{\theta}/dt = -\Gamma\, Y^T s$ garantisce $s \to 0$ e $\tilde{\theta} := \hat{\theta} - \theta$ limitati per ogni condizione iniziale $(\tilde{\eta}(0), \dot{s}(0), \hat{\theta}(0))$.

Dimostrazione. Si adotta la funzione di Lyapunov composta $V = \tfrac{1}{2}\, s^T M(\eta)\, s + \tfrac{1}{2}\, \tilde{\theta}^T \Gamma^{-1} \tilde{\theta}$. La derivata temporale è $\dot{V} = s^T M \dot{s} + \tfrac{1}{2}\, s^T \dot{M} s + \tilde{\theta}^T \Gamma^{-1} \dot{\tilde{\theta}}$. Sostituendo le equazioni del moto con il controllore adattivo e la legge di aggiornamento: $\dot{V} = s^T(Y\theta - Y\hat{\theta} - K_D s) + \tfrac{1}{2}\, s^T \dot{M} s + \tilde{\theta}^T \Gamma^{-1}(-\Gamma Y^T s) = -s^T K_D s + s^T Y \tilde{\theta} - \tilde{\theta}^T Y^T s + \tfrac{1}{2}\, s^T (\dot{M} - 2C)\, s$. Il termine antisimmetrico $(\dot{M} - 2C)$ annulla il quadratic form (Proposizione 1.1), e $s^T Y \tilde{\theta} - \tilde{\theta}^T Y^T s = 0$ per l'identità $s^T Y\tilde{\theta} = \tilde{\theta}^T Y^T s$ (scalare). Quindi $\dot{V} = -s^T K_D s \le 0$. Dall'invarianza di LaSalle si conclude $s \to 0$. Con persistente eccitazione, $\tilde{\theta} \to 0$. □

Integrazione con il WBC gerarchico

L'integrazione del controllo adattivo nel WBC gerarchico (§2.2) richiede cautela: l'aggiornamento dei parametri $\hat{\theta}$ modifica $M(\eta)$ e $h(\eta,\dot{\eta})$ ad ogni ciclo di controllo, rendendo il QP gerarchico dipendente dallo stato adattivo. La soluzione standard è usare un aggiornamento a due scale temporali: i parametri $\hat{\theta}$ vengono aggiornati a bassa frequenza (50–100 Hz), mentre il QP del WBC opera a 500–1000 Hz con parametri congelati sull'intervallo. Questa separazione delle scale è giustificata dalla lentezza tipica della deriva dei parametri inerziali rispetto alla dinamica del controllo.

Capitolo 2.7 — Controllo Adattivo di Forza e Ammettenza per la Manipolazione a Contatto

Raccordo con la monografia: § 2.2–2.4 (WBC, task-space controller, stabilita).

Il Whole-Body Controller del Capitolo 2 e progettato per il controllo di posizione nello spazio dei task: l'obiettivo e raggiungere e mantenere una posa target T_d in SE(3). Quando il robot entra in contatto con l'ambiente -- durante la manipolazione, l'assemblaggio, o l'assistenza fisica -- il controllo di posizione puro e inadeguato: rigidita eccessiva provoca forze di contatto non controllate che possono danneggiare oggetti fragili o destabilizzare il robot. Il controllo di impedenza e ammettenza risolve questo problema regolando dinamicamente la relazione forza-spostamento all'interfaccia robot-ambiente.

2.1 Modello di Impedenza Meccanica

Il controllo di impedenza (Hogan 1985) specifica la relazione dinamica desiderata tra l'errore di posizione e la forza di contatto come un sistema massa-molla-smorzatore virtuale:

$$M_d * x_{ddot_e} + D_d * x_{dot_e} + K_d * x_e = F_{ext}$$

dove x_e = x - x_d e l'errore di posizione cartesiano, M_d, D_d, K_d sono rispettivamente inerzia, smorzamento e rigidezza virtuali desiderati, e F_ext e la forza esterna misurata ai polsi. La legge di controllo che realizza questa equazione in task-space e:

$$tau = Y(eta,eta_{dot},eta_{ddot_r}) pi + J^T (F_d - K_d x_e - D_d x_{dot_e})$$

dove F_d e la forza di feedforward desiderata e eta_ddot_r = x_ddot_d - M_d^{-1}(D_d x_dot_e + K_d x_e) e l'accelerazione di riferimento. La scelta di M_d, D_d, K_d determina la "morbidezza" del comportamento al contatto: K_d piccola produce un comportamento cedevole (soft contact), K_d grande riproduce il controllo di posizione rigida.

2.2 Controllo di Ammettenza: Dualita con l'Impedenza

Il controllo di ammettenza e duale al controllo di impedenza: invece di specificare la relazione forza->spostamento (impedenza), specifica la relazione spostamento->forza (ammettenza). Dato un riferimento di forza F_d e la forza misurata F_meas, il controllo di ammettenza genera una correzione di posizione:

$$M_a * delta_{x_ddot} + D_a * delta_{x_dot} + K_a * delta_x = F_d - F_{meas}$$

La posizione di riferimento modificata x_ref = x_d + delta_x viene poi inseguita dal WBC con un controller di posizione rigido. Questo approccio e preferibile quando il robot e in posizione di forza -- ad esempio durante l'assemblaggio -- perche il controller di posizione interno garantisce la precisione di

posizionamento, mentre il loop di ammettenza esterno gestisce la conformita alla forza.

2.3 Controllo di Ammettenza Adattivo con Stima del Payload PINN

Proposizione S2.1 (Stabilita del controllo di ammettenza con parametri variabili).

Sia M_a(pi_hat) la matrice di inerzia virtuale parametrizzata dalla stima PINN pi_hat del payload. Se la stima converge || pi_hat(t) - pi || -> 0 esponenzialmente con tasso alpha_id, allora il sistema chiuso ammettenza + PINN e asintoticamente stabile con margine di stabilita proporzionale ad alpha_id.

Dimostrazione. Si adotta la funzione di Lyapunov composta V = V_amm(x_e, x_dot_e) + V_id(pi_tilde) dove V_amm e la funzione di Lyapunov del sistema di ammettenza e V_id = (1/2) pi_tilde^T Gamma^{-1} pi_tilde e quella dell'identificatore (Teorema 2.3, controllore adattivo di Slotine-Li). La derivata V_dot <= -c1 ||x_e||^2 - c2 ||pi_tilde||^2 e negativa definita per guadagni sufficientemente grandi. []

2.4 Applicazione: Assemblaggio Preciso con Feedback di Forza

Il controllo di ammettenza e la tecnica di elezione per task di peg-in-hole (inserimento di componenti in fori di tolleranza ridotta). L'algoritmo tipico combina tre fasi: (1) ricerca della posizione del foro con spirale di forza, (2) inserimento con controllo di ammettenza che compensa le forze di attrito laterali, (3) verifica della profondita di inserimento tramite la diminuzione della forza assiale. La matrice di ammettenza ottimale per questa applicazione ha K_a alta lungo l'asse di inserimento (Z) e bassa nelle direzioni trasversali (X, Y), permettendo aggiustamenti laterali fluidi mentre si mantiene la progressione lungo Z.

Capitolo 3 — Locomozione Bipede su Terreni non Strutturati

La locomozione bipede in ambienti reali presenta una complessità qualitativa che trascende quella della camminata in linea retta su superficie piana, scenario sul quale la maggioranza dei risultati teorici è stata sviluppata. Il terreno non strutturato — gradini, rampe, terreni cedevoli, ostacoli, superfici sconnesse — impone al sistema di controllo la capacità di pianificare in tempo reale la sequenza dei passi, adattare la dinamica del corpo durante la fase di volo e gestire impatti con geometrie di contatto non predeterminate.

3.1 Il Modello dell'Inverted Pendulum su Carrello

Il punto di partenza di quasi ogni teoria della locomozione bipede è il modello lineare dell'inverted pendulum su carrello (LIPM), che idealizza il robot come una massa puntiforme posta all'estremità di un pendolo di lunghezza costante z_c. Le equazioni del moto del CoM si riducono a:

$$\ddot{x} = \omega^2 0\ (x - z\_\{COP\}),\ \omega_0 = \sqrt{\frac{g}{z\_c}} \quad (3.1)$$

dove z_COP è la coordinata orizzontale del Centro di Pressione. La soluzione generale si scrive come x(t) = x⁺ e^{ω₀t} + x⁻ e^{−ω₀t} + z_COP. Definendo la Capture Point come ξ = x + ẋ/ω_0, la sua dinamica è ξ̇ = ω_0(ξ − z_COP): il CP è un integratore del segnale (ξ − z_COP), e per rendere il sistema stabile è sufficiente scegliere z_COP = ξ + k(ξ − ξ^ref) con k > 0.

3.2 Pianificazione dei Passi come Ottimizzazione su Varietà

Sia la sequenza dei passi (p_k, R_k, t_k) ∈ SE(2) × ℝ_{>0}, dove p_k = (x_k, y_k)⊤ è la posizione del piede k-esimo, R_k ∈ SO(2) la sua orientazione e t_k il tempo di atterraggio. Il problema di footstep planning può essere formulato come un programma misto-intero quadratico (MIQP) che minimizza la deviazione dalla direzione nominale e lo sforzo di controllo:

$$min\ _{\{p\_k,\ t\_k\}} \sum\ _k \left[\| p\ \ _k - p^{nom}\ \ _k \|^2\ \ _{W\_p} + w\_t(t\_k - T\_s)^2 \right] \quad (3.2)$$

soggetto ai vincoli cinematici di raggiungibilità, ai vincoli di non collisione tra i piedi e ai vincoli di stabilità dinamica espressi in termini della condizione di cattura. La struttura di questo problema è quella di un MILP in molti casi pratici, se la funzione di costo è riformulata in norma ℓ_1 o $\ell\infty$ e i politopi di raggiungibilità sono descritti esplicitamente.

3.3 Sistemi Ibridi e Mappa di Poincaré

La locomozione può essere modellata come un sistema dinamico ibrido con resets (HDSR). Gli eventi discreti sono i contatti con il suolo, modellati come guardie $\Phi_i(q) \leq 0$ che definiscono superfici in TQ; le leggi di reset modellano l'impatto come una mappa discontinua delle velocità articolari. Per un robot bipede che cammina in modo periodico, si definisce la sezione di Poincaré $\Sigma \subset TQ$ — tipicamente lo stato immediatamente dopo il touchdown — e si studia la mappa di ritorno $P: \Sigma \to \Sigma$. Un ciclo limite corrisponde a un punto fisso x^* con $P(x^*) = x^*$, e la sua stabilità locale è determinata dagli autovalori di $DP(x^*)$: il ciclo è asintoticamente stabile se e solo se tutti gli autovalori di $DP(x^*)$ hanno modulo strettamente inferiore all'unità.

3.3+ Teoria degli Hybrid Zero Dynamics (HZD)

La teoria degli Hybrid Zero Dynamics, sviluppata da Westervelt e Grizzle nel corso degli anni 2000 e culminata nella monografia del 2007, fornisce il quadro matematico più rigoroso disponibile per l'analisi e la sintesi di controllori per robot bipedi che camminano in modo periodico. L'idea fondamentale è ridurre il sistema ibrido ad alta dimensionalità a un sistema a dimensione ridotta — le zero dynamics — la cui stabilità implica quella dell'intero ciclo di camminata.

3.3.1 Il Sistema Ibrido Continuo-Discreto

Consideriamo un robot bipede con n gradi di libertà articolari e corpo base floating. La configurazione $q \in Q \subseteq \mathbb{R}^n$ descrive completamente la geometria del sistema. Il sistema dinamico ibrido è definito dalla coppia (Σ_C, Δ), dove Σ_C è il sistema continuo (equazioni di Eulero-Lagrange durante la fase di singolo appoggio) e Δ è la mappa di reset che modella l'impatto del piede swing.

La fase di singolo appoggio è governata dalle equazioni del moto lagrangiano con vincoli di contatto:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu + J_{\text{◡}}^{T}(q)\lambda \quad (3.3.1)$$

dove $D(q) \in \mathbb{R}^{n \times n}$ è la matrice di inerzia, $C(q,\dot{q})$ contiene i termini di Coriolis e centrifughi, $G(q)$ è il gradiente dell'energia potenziale, $B \in \mathbb{R}^{n \times m}$ è la matrice di mappatura degli attuatori, $u \in \mathbb{R}^{m}$ è il vettore di coppia, $J_{\text{◎}}(q)$ è la Jacobiana dei vincoli di contatto stazionario e λ sono i moltiplicatori di Lagrange associati.

L'evento di impatto è definito dalla guardia $S = \{(q,\dot{q}) \in TQ \mid h_{\text{◎}}(q) = 0, \dot{h}_{\square}(q,\dot{q}) < 0\}$, dove $h_{\text{◎}}(q) = 0$ è la superficie di atterraggio del piede swing. La mappa di reset $\Delta: S \to TQ$ modella l'impatto come anelastico:

$$(q^{+}, \dot{q}^{+}) = \Delta(q^{-}, \dot{q}^{-}) = (R_q\, q^{-}, R_{\dot{q}}(q^{-})\, \dot{q}^{-}) \quad (3.3.2)$$

dove R_q è la permutazione che scambia le gambe e $R_{\dot{q}}(q^{-})$ è la mappa di conservazione del momento angolare — derivata dalla condizione che, durante l'impatto istantaneo, le forze di impatto producano un impulso ma il momento risultante rispetto al nuovo punto di contatto sia nullo. L'espressione esplicita è:

$$R_{\dot{q}}(q^{-}) = D(q^{+})^{-1}\,[D(q^{-}) - J_a^{T}(q^{-})(J_a D^{-1} J_a^{T})^{-1} J_a(q^{-})] \cdot D(q^{-})^{-1}\, D(q^{-}) \quad (3.3.3)$$

dove $J_a(q)$ è la Jacobiana del piede che atterra valutata nella configurazione pre-impatto. La derivazione si ottiene imponendo la condizione che il momento angolare rispetto al nuovo contatto sia conservato e che la velocità del piede nel punto di contatto si annulli istantaneamente.

3.3.2 Output Virtuali e Parametrizzazione della Camminata

Il cuore della teoria HZD è la nozione di output virtuale: un insieme di m funzioni $y_i(q)$ che il controllore porta a zero, definendo così una sottovariante del sistema a dimensione ridotta. La scelta degli output virtuali encode la forma desiderata della camminata.

Sia $\theta(q) \in \mathbb{R}$ una coordinata di fase virtuale — tipicamente l'angolo del tronco rispetto alla verticale, o una funzione delle coordinate articolari che

è strettamente monotona durante la camminata. Gli output virtuali sono definiti come:

$$y = h(q) = q\_a - h\_d(\theta(q)) \quad (3.3.4)$$

dove q_a $\in \mathbb{R}^m$ sono le coordinate articolari attuate e h_d: $\mathbb{R} \to \mathbb{R}^m$ è la traiettoria desiderata parametrizzata dalla coordinata di fase θ. La forma usuale per h_d è una curva di Bézier:

$$h\_d(\theta) = \Sigma_{i=0}^{N} \alpha_i \, b_{i,N}(s(\theta)), \quad s(\theta) = (\theta - \theta^-)/(\theta^+ - \theta^-) \quad (3.3.5)$$

dove $b_{i,N}$ sono i polinomi di base di Bernstein di grado N e $\alpha_i \in \mathbb{R}^m$ sono i coefficienti da ottimizzare. La variabile normalizzata $s(\theta) \in [0,1]$ mappa l'intervallo $[\theta^-, \theta^+]$ dell'angolo di fase nell'intero ciclo di passo.

Il controllore di feedback che porta $y \to 0$ è un PD basato sulla dinamica dell'output:

$$u^* = -(L\_f^2 h + (A_0 y + A_1 \dot{y})) / L\_g \, L\_f \, h \quad (3.3.6)$$

dove *Lf* e *Lg* sono derivate di Lie lungo i campi vettoriali f (dinamica libera) e g (effetto degli attuatori), e A_0, A_1 sono matrici di guadagno scelte affinché il polinomio caratteristico $s^2 + A_1 s + A_0$ abbia radici con parte reale negativa. Questo assicura la convergenza esponenziale di y verso zero.

3.3.3 La Varietà Zero e le Zero Dynamics

Quando il controllore (3.3.6) è in azione e il sistema è sul manifold $y = 0$, $\dot{y} = 0$, la dinamica residua è quella delle zero dynamics, governata dalle coordinate non attuate. Per un sistema correttamente relativamen definito, la dimensione delle zero dynamics è $n - m$, dove m è il numero di output virtuali e n il numero totale di gradi di libertà.

Il teorema fondamentale delle HZD afferma che, per il sistema continuo in fase di singolo appoggio, la varietà zero $Z = \{(q,\dot{q}) \mid y = 0, \dot{y} = 0\}$ è invariante sotto il flusso del sistema in anello chiuso — cioè, se le condizioni iniziali sono in Z, le traiettorie rimangono in Z. Questo riduce lo studio della stabilità del ciclo limite a un sistema di dimensione ridotta.

Le zero dynamics sulla varietà Z sono un sistema del secondo ordine in θ, la coordinata di fase:

$$\sigma(\theta)\ddot{\theta} + (1/2)(d\sigma/d\theta)\dot{\theta}^2 = \rho(\theta) \quad (3.3.7)$$

dove $\sigma(\theta) = (\partial\theta/\partial q)\, D^{-1}(\partial\theta/\partial q)^T$ è la inerzia virtuale (uno scalare positivo lungo Z) e $\varrho(\theta) = -(\partial\theta/\partial q)\, D^{-1}\, g(q)|_{Z}$ è il termine di forza generalizzato proiettato sulla direzione θ. Questo sistema del secondo ordine ha l'interpretazione fisica di un pendolo inverso parametrico con inerzia variabile.

> ▸ *Osservazione: La riduzione alle zero dynamics è analoga al teorema di riduzione di Routh per sistemi con simmetrie cicliche, ma si applica qui a sistemi vincolati su sottovarianti.*

3.3.4 Costruzione della Mappa di Poincaré Ibrida

La mappa di Poincaré per il sistema ibrido HZD si costruisce in modo rigoroso a partire dalla composizione del flusso continuo con la mappa di reset. Definiamo la sezione di Poincaré come la sottovariante della varietà zero all'evento di impatto:

$$\Sigma_Z = Z \cap S = \{(q,\dot{q}) \in Z \mid h_{\text{swing}}(q) = 0,\ \dot{h}_{\text{swing}}(q,\dot{q}) < 0\} \quad (3.3.8)$$

Poiché Z ha dimensione $n - m$ e $\Sigma \subset TQ$ è un'ipersuperficie (codimensione 1), la sezione Σ_Z ha dimensione $n - m - 1$. Per un robot bipede minimale (sistemi planari semplici come RABBIT o ERNIE), con $n = 5$, $m = 4$ output virtuali, si ottiene una sezione 1-dimensionale — riducendo la mappa di Poincaré a una funzione scalare.

La mappa di Poincaré $P: \Sigma_Z \to \Sigma_Z$ è definita come composizione:

$$P = (\Phi_{T_Z})|_{\Sigma_Z} \circ \Delta_Z \quad (3.3.9)$$

dove $\Delta_Z = P_{Z^+} \circ \Delta$ è la restrizione della mappa di reset alla varietà zero dopo l'impatto (P_{Z^+} è la proiezione sulla varietà zero post-impatto), e Φ_{T_Z} è il flusso delle zero dynamics integrato fino al successivo evento di impatto.

In coordinate locali, con $x = \theta|_{\Sigma_Z}$ come coordinata sulla sezione 1-dimensionale, la mappa di Poincaré diventa la mappa scalare $x^+ = P(x)$, dove $\dot{x}$ è la velocità della coordinata di fase al momento del touchdown. Un ciclo di camminata periodica corrisponde a un punto fisso $x^* = P(x^*)$.

3.3.5 Analisi della Stabilità Orbitale

La stabilità del ciclo limite corrispondente al punto fisso x^* è determinata dal valore proprio (eigenvalue) della differenziale $DP(x^*) = dP/dx|_{x=x^*}$. Il ciclo è:

- **Asintoticamente stabile** se $|DP(x^*)| < 1$ (il punto fisso è un attrattore)
- **Marginalmente stabile** se $|DP(x^*)| = 1$ (comportamento non generico)
- **Instabile** se $|DP(x^*)| > 1$ (il punto fisso è un repulsore)

Per sistemi multidimensionali (sezione di Poincaré di dimensione $n - m - 1 > 1$), la condizione di stabilità asintotica generalizza a: tutti gli autovalori della matrice Jacobiana $DP(x^*) \in \mathbb{R}^{(n-m-1)\times(n-m-1)}$ devono avere modulo strettamente minore di uno. Questi autovalori misurano il rateo di convergenza delle perturbazioni trasverse all'orbita periodica.

Il calcolo numerico di $DP(x^*)$ si esegue tramite differenziazione numerica della mappa P — perturbando le condizioni iniziali attorno a x^* e misurando la deviazione dopo un passo. Per sistemi semplici, è anche possibile una derivazione analitica sfruttando la linearizzazione delle equazioni di variazione lungo l'orbita nominale.

> ▸ *Connessione con Capitolo 2: La condizione $|DP(x^*)| < 1$ equivale alla contrazione del sistema linearizzato attorno al ciclo. Questo è il criterio di stabilità di Lyapunov per mappe discrete, analogo alla stabilità asintotica per sistemi ODEontinui — si confronti con la condizione $Re_i(A_i) < 0$ per i sistemi continui trattati nel Capitolo 2.*

3.3.6 Il Teorema di Grizzle sulla Coordinata di Fase Virtuale

Il teorema di Grizzle sulla sincronizzazione della camminata tramite coordinata di fase virtuale è uno dei risultati centrali della teoria HZD. Risolve il problema della sincronizzazione dei gradi di libertà del robot in assenza di un segnale di clock esterno.

Teorema (Grizzle, 2003): Sia $\theta(q)$ una coordinata di fase virtuale strettamente monotona lungo la traiettoria nominale di camminata ($\dot{\theta} > 0$ lungo il ciclo). Supponiamo che le zero dynamics con rispetto θ ammettano un ciclo limite orbitalmente stabile con punto fisso x^* sulle sezione Σ_Z. Allora, per ogni condizione iniziale in un intorno tubulare U

dell'orbita nominale nella varietà Z, il sistema HZD in anello chiuso converge esponenzialmente verso il ciclo limite.

La dimostrazione sfrutta tre ingredienti:

(1) La monotonia di θ garantisce che θ possa essere usata come coordinata tempificante lungo il ciclo — ogni parte del corpo del robot è, in linea di principio, in fase con ogni altra tramite la coordinata condivisa θ. Questo elimina la necessità di un timer esplicito e rende la policy invariante rispetto a perturbazioni temporali.

(2) La stabilità orbitale della mappa di Poincaré garantisce che le perturbazioni trasverse all'orbita decadano esponenzialmente nel senso della topologia del ciclo.

(3) La connessione tra le due strutture — la convergenza verso Z e la stabilità orbitale all'interno di Z — si combina in modo algebrico preciso tramite il lemma di Lagrange per sistemi ibridi (o lemma di Grizzle-Chevallereau-Plestan), che afferma: se il ciclo limite delle zero dynamics è asintoticamente stabile con tasso λ_ZD < 1, e l'output y converge verso zero con tasso λ_y < 1, allora il sistema completo è asintoticamente stabile con tasso λ_totale = max(λ_ZD, λ_y).

$$\|(q(t), \dot{q}(t)) - (q\_nom(t), \dot{q}\_nom(t))\| \leq C\, e^{\{-\alpha t\}}\, \|(q(0), \dot{q}(0)) - (q\_nom(0), \dot{q}\_nom(0))\| \quad (3.3.10)$$

per opportune costanti C > 0 e α > 0 dipendenti dai guadagni del controllore e dai parametri meccanici del robot.

3.3.7 Collegamento con la Sezione 3.2: Ottimizzazione dei Coefficienti di Bézier

I parametri α_i delle curve di Bézier in (3.3.5) non sono arbitrari: vengono determinati risolvendo un problema di ottimizzazione vincolata che impone la periocità del ciclo di camminata e la stabilità delle zero dynamics. Il problema di ottimizzazione prende la forma:

$$\min\_\{\alpha\}\ \Sigma_i\, w_i\, J_i(\alpha) \quad s.t. \quad x^* = P(x^*; \alpha), \quad |DP(x^*; \alpha)| < 1 - \varepsilon, \quad Vincoli\ fisici \quad (3.3.11)$$

dove $J_i(\alpha)$ sono obiettivi multipli (efficienza energetica, robustezza, comfort del passo), $x^* = P(x^*; \alpha)$ è il vincolo di periodicità, $|DP(x^*; \alpha)| < 1 - \varepsilon$ è il vincolo di stabilità (con margine $\varepsilon > 0$), e i vincoli fisici includono limiti sulle coppie, sulle velocità articolari e sulla clearance del piede.

Questo collega la sezione 3.2 (pianificazione come ottimizzazione) con la sezione 3.3 (stabilità ibrida): il footstep planning ottimizza la traiettoria di riferimento, mentre l'HZD garantisce che il sistema la segua in modo stabile in presenza di perturbazioni. La coordinata di fase virtuale $\theta(q)$ funge da anello di congiunzione tra i due livelli di astrazione.

> ▸ *Connessione con il Capitolo 2: Il problema (3.3.11) è un'istanza di ottimizzazione non lineare vincolata su una varietà, analogo alla programmazione semidefinita (SDP) usata nella sintesi dei controllori LQR con vincoli di stabilità. I metodi di punto interno o di shooting multiplo sono tipicamente impiegati per la sua soluzione numerica.*

3.4 Il modello SLIP e la dinamica della corsa bipede

Il modello SLIP (Spring-Loaded Inverted Pendulum), proposto originalmente da Blickhan nel 1989 come descrizione biomeccanica della corsa degli animali bipedi, è il modello a parametri concentrati più semplice capace di riprodurre le caratteristiche meccaniche fondamentali del ciclo di corsa: la fase di stance con accumulo e rilascio di energia elastica, la fase di volo con traiettoria balistica, e la transizione discontinua tra le due. Nonostante la sua semplicità apparente, il SLIP è matematicamente non integrabile in forma chiusa: la dinamica di stance è non lineare e le sue soluzioni richiedono integrazione numerica o approssimazioni analitiche.

Il modello descrive il corpo come una massa puntiforme m posta all'estremità di un arto elastico di lunghezza a riposo l_0 e costante elastica k, privo di massa. In coordinate polari (r, θ) centrate sull'appoggio del piede, le equazioni del moto durante la fase di stance sono:

$$m\ddot{r} - mr\dot{\theta}^2 = -mg\cos\theta + k(l_0 - r)$$

$$m(r\ddot{\theta} + 2\dot{r}\dot{\theta}) = mg\sin\theta$$

La prima equazione governa la dinamica radiale (compressione-estensione della molla); la seconda governa la rotazione attorno al punto di appoggio. L'energia meccanica totale durante la fase di stance è:

$$E = \tfrac{1}{2}\, m(\dot{r}^2 + r^2\dot{\theta}^2) + mgr\cos\theta + \tfrac{1}{2}\, k(l_0 - r)^2$$

ed è un integrale primo del moto, poiché il sistema è conservativo. Tuttavia, il momento angolare rispetto al punto di appoggio non è conservato (la forza elastica ha componente tangenziale non nulla quando $r \neq l_0$ e $\theta \neq \pi/2$), e non esiste un secondo integrale primo in forma chiusa, rendendo il sistema non hamiltonianamente integrabile.

La fase di volo ha durata finita e traiettoria puramente balistica: l'angolo di attacco del piede al successivo touchdown — denominato angle of attack α_0 — è il parametro di controllo del ciclo. Studi numerici sistematici (Geyer, Seyfarth, Blickhan 2005) mostrano che, per valori fisicamente realistici dei parametri ($k/mg \approx 10$–20, lunghezza normalizzata $l_0/h \approx 0{,}9$ dove h è l'altezza del CoM), la mappa di Poincaré P del modello SLIP ha una ricca struttura: esiste una famiglia di punti fissi corrispondenti alla corsa periodica, la cui stabilità dipende criticamente dall'angolo di attacco α_0. La condizione di marginalità — autovalore unitario della linearizzazione di P — definisce una curva nel piano (E, α_0) che separa la regione di corsa stabile da quella instabile.

L'approssimazione analitica di Spring-Mass SLIP, valida per compressioni $\varepsilon = 1 - r\_min/l_0$ piccole, si ottiene perturbando attorno all'orbita simmetrica. A primo ordine in ε, il punto di rimbalzo r_min soddisfa:

$$\tfrac{1}{2}\, k\,(l_0 - r\_min)^2 + mg\, r\_min\cos\theta\_\{td\} = E\_stance, \quad con \quad E\_stance = E - \tfrac{1}{2}\, k\, l_0^2$$

(termine di volo)

e l'impulso della forza di reazione al suolo nella direzione verticale è, nell'approssimazione di arto rigido a piccoli angoli, proporzionale a $k\,(l_0 - r\_min) / \omega\_n$ con $\omega\_n = \sqrt{(k/m)}$. Questo consente di calcolare analiticamente la variazione del CoM orizzontale per passo, fornendo una stima del range di velocità di corsa sostenibile in funzione dei parametri meccanici.

Hybrid Zero Dynamics: Costruzione e Proprietà

Un bipede meccanico durante un ciclo di camminata alterna tra due fasi: la **fase di singolo appoggio** (single support, SS), in cui un piede è a contatto con il suolo e il sistema evolve secondo la dinamica continua, e la **fase di doppio appoggio** (double support, DS), che per i modelli idealizzati si riduce a un **impatto istantaneo** quando il piede oscillante tocca terra. Il sistema complessivo è ibrido:

$$\Sigma_{hybrid} = \{\dot{x} = f(x) + g(x)u \; fase \; SS \; x^{+} = \Delta\left(x^{-}\right) \; impatto$$

dove $x = \left(q, \dot{q}\right) \in TQ$ è lo stato, $u \in R^{m}$ il controllo, e $\Delta: S \to TQ$ è la mappa di impatto definita sulla surface of section $S = \{x: \phi_{n}(q) = 0, \dot{\phi}_{n}\left(q, \dot{q}\right) < 0\}$.

Variabile di Fase Virtuale e Vincoli di Output

Il cuore della teoria HZD è la **variabile di fase** virtuale $\theta(q): Q \to [0, 2\pi)$ che parametrizza il progresso del ciclo di camminata senza fare riferimento al tempo esplicito. Tipicamente θ è definita come una funzione della posizione del tronco o dell'angolo della gamba di stance. I **vincoli di output virtuali** sono funzioni $y: TQ \to R^{m}$ della forma:

$$y_{k}(q) = q_{k} - h_{k}(\theta(q)), \quad k = 1, \ldots, m$$

dove $h_{k}: [0, 2\pi) \to R$ è una curva di Bézier che specifica la traiettoria desiderata del giunto k in funzione della fase θ. La **varietà zero** $Z = \{x \in TQ: y(x) = 0, \dot{y}(x) = 0\}$ è il sottospazio invariante su cui si sviluppa la dinamica ridotta del bipede.

Definizione 3.1 (Hybrid Zero Dynamics). *La dinamica del sistema* Σ_{hybrid} *ristretta alla varietà zero* Z *è chiamata Hybrid Zero Dynamics (HZD). Essa è una mappa di Poincaré unidimensionale:*

$$P_{Z}: Z \cap S \to Z \cap S$$

che descrive l'evoluzione della variabile di fase θ *e della sua derivata* $\dot{\theta}$ *da un impatto al successivo.*

Teorema 3.1 (Invarianza di Z sotto il controllo HZD, Westervelt et al. 2003). *Sia* $u^* = u^*(x)$ *il controllore che impone* $\ddot{y} = -K_p y - K_d \dot{y}$ *con* $K_p, K_d > 0$ *e i vincoli di output* y *differenziabili. Se* $y(x_0) = 0$ *e* $\dot{y}(x_0) = 0$, *allora* $y(x(t)) = 0$ *e* $\dot{y}(x(t)) = 0$ *per ogni* $t \geq 0$ *nella fase SS, a condizione che lo Jacobiano* $\partial y/\partial q$ *abbia rango* m.

Dimostrazione. Il controllore PD su y rende Z invariante per il sistema in fase SS: l'equazione $\ddot{y} = -K_p y - K_d \dot{y}$ con condizioni iniziali $y(0) = 0, \dot{y}(0) = 0$ ha soluzione $y(t) \equiv 0$ per $K_p, K_d > 0$. L'invarianza durante la fase SS è quindi garantita per costruzione. La condizione di rango pieno assicura che l'equazione $\ddot{y} = u_y$ possa essere soddisfatta tramite la scelta appropriata di u^*. ▫

Mappa di Poincaré e Analisi della Stabilità Orbitale

La stabilità di un ciclo di camminata periodico nell'HZD è analizzata tramite la **mappa di Poincaré** $P: S \cap Z \to S \cap Z$. Un punto fisso $x^* = P(x^*)$ corrisponde a un ciclo di camminata periodico stabile.

La stabilità locale del ciclo è determinata dalla **linearizzazione della mappa di Poincaré** $DP(x^*) \in R^{(2n-m)\times(2n-m)}$ (dove $2n - m$ è la dimensione di $Z \cap S$). Il ciclo è: - **Localmente asintoticamente stabile** se tutti gli autovalori di $DP(x^*)$ hanno modulo minore di 1. - **Instabile** se esiste almeno un autovalore con modulo maggiore di 1.

Teorema 3.2 (Linearizzazione della mappa di Poincaré per HZD). *La mappa di Poincaré dell'HZD ha derivata:*

$$DP(x^*) = \left[\frac{\partial \Delta}{\partial x}\Big|_{x^-} \cdot \Phi(T; x^*) \right]_Z$$

dove $\Phi(T; x^*)$ *è la matrice di transizione fondamentale del sistema lineare variante nel tempo ottenuto linearizzando* $\dot{x} = f(x) + g(x)u^*(x)$ *lungo il ciclo nominale* $x^*(t)$ *di periodo* T, *e* $[\cdot]_Z$ *denota la restrizione alla varietà tangente di* Z.

Dimostrazione. Segue dalla teoria dei sistemi ibridi periodici. La derivata della mappa di Poincaré si ottiene componendo: (1) la derivata della

dinamica continua (matrice di transizione), (2) la derivata della proiezione sul piano di Poincaré, e (3) la derivata della mappa di impatto $\partial\Delta/\partial x$. La restrizione a Z riduce la dimensione da $2n$ a $2n - m$ tramite la proiezione sul null-space dei vincoli di output. ▫

3.5 Hybrid Zero Dynamics e controllo orbitale della camminata

La teoria delle Hybrid Zero Dynamics (HZD), sviluppata da Westervelt, Grizzle e collaboratori nel corso del 2000-2007, rappresenta il contributo più rigoroso alla sintesi di controllori per robot bipedi che camminano in modo periodico. L'idea fondamentale è di progettare una legge di controllo che rende invariante una sottovarietà bidimensionale dello spazio degli stati — la superficie di zero dynamics — e di analizzare la stabilità del ciclo limite su tale sottovarietà.

Si considera un robot bipede con n gradi di libertà articolari, rappresentato come sistema ibrido con due fasi: la fase di singolo supporto (SS, un grado di libertà del piede di appoggio è in contatto col suolo) e l'evento di impatto al touchdowndel piede libero. Durante la fase SS, la dinamica è:

$$M(q)\,\ddot{q} + C(q,\dot{q})\,\dot{q} + G(q) = B\,u$$

dove u ∈ ℝ^{n−1} è il vettore delle coppie articolari e B ∈ ℝ^{n×(n−1)} è la matrice di attuazione (il piede di appoggio non è attuato). Il sistema ha un grado di libertà non attuato, il che rende il problema di controllo fondamentalmente diverso da quello del manipolatore: non si può imporre una traiettoria arbitraria a tutti i gradi di libertà.

Si scelgono m = n − 1 variabili di output y = h(q) ∈ ℝ^m, tipicamente le posizioni dei giunti relativamente al giunto di stance (angolo delle anche, ginocchia, caviglie) parametrizzate rispetto all'angolo θ della gamba di supporto (coordinata di fase). La scelta della parametrizzazione rispetto a θ — invece che al tempo — è essenziale: rende il design del controllore invariante rispetto alla velocità di avanzamento e consente di usare polinomi di Bézier come curve di riferimento.

Il controllore di zeroing degli output è progettato per rendere $y \to 0$ e $\dot{y} \to 0$ in tempo finito (sliding mode) o asintoticamente (linearizzazione esatta del feedback). Con linearizzazione esatta del feedback:

$$u = (L_g\, L_f\, h(q))^{\{-1\}}\, [\ddot{y}_d - L_f^2\, h(q) - K_p\, h(q) - K_d\, L_f\, h(q)]$$

dove L_f e L_g sono le derivate di Lie lungo i campi f (dinamica libera) e g (attuazione), rispettivamente. Questa legge di controllo richiede che L_g L_f h(q) sia invertibile, condizione che si verifica quasi ovunque lungo le traiettorie nominali ed è strettamente legata alla struttura di accoppiamento ingresso-uscita del sistema. Una volta applicato il controllore di zeroing, il sistema è costretto sulla sottovarietà invariante $S = \{(q, \dot{q}) : h(q) = 0,\ L_f\, h(q) = 0\} \subset TQ$, denominata zero dynamics manifold. La dimensione di S è 2 (dim TQ − 2m) = 2: la coordinata θ e la sua velocità $\dot{\theta}$ bastano a descrivere completamente il moto sul manifold. Le zero dynamics sono la restrizione del sistema a S:

$$\ddot{\theta} = \varphi(\theta, \dot{\theta}) \quad \textit{(funzione scalare)}$$

dove $\varphi(\theta, \dot{\theta})$ è determinata sostituendo le espressioni h(q) = 0 nell'equazione del moto non attuata. Poiché il sistema è energeticamente conservativo sulla varietà di zero dynamics (nell'approssimazione adiabatic), esiste una funzione di energia del ciclo:

$$E(\theta, \dot{\theta}) = \tfrac{1}{2}\, D(\theta)\, \dot{\theta}^2 + V_z(\theta)$$

dove $D(\theta) > 0$ è l'inerzia scalare e $V_z(\theta)$ è il potenziale effettivo sulle zero dynamics. La camminata periodica corrisponde a una traiettoria chiusa nella sezione di Poincaré $\Sigma = \{\theta = \theta_f,\ \dot{\theta} > 0\}$, cioè a un punto fisso $(\theta_f, \dot{\theta}_f^*)$ per la mappa di ritorno $P: \Sigma \to \Sigma$ definita dal ciclo ibrido (zero dynamics + legge di impatto + reset delle variabili di stato al touchdown).

> **Teorema 3.1 (Stabilità della camminata HZD).** *Sia $(\theta^*, \dot{\theta}^*)$ un punto fisso della mappa di Poincaré P e sia $\rho(DP) = \max_i |\lambda_i(DP(\theta^*, \dot{\theta}^*))|$ il raggio spettrale della sua linearizzazione. Il ciclo limite della camminata è orbitalmente asintoticamente stabile se e solo se $\rho(DP) < 1$.*

Dimostrazione. La condizione ϱ(DP) < 1 è la definizione di punto fisso asintoticamente stabile per una mappa discreta. L'invarianza del manifold S sotto il flusso e la regola di impatto garantisce che la stabilità orbitale (convergenza a un intorno della traiettoria periodica) equivale alla stabilità del punto fisso della mappa di Poincaré ristretta allo spazio bidimensionale (θ, θ̇). La continuità di P rispetto alle condizioni iniziali e al parametro α_0 di controllo consente di ottimizzare i coefficienti dei polinomi di Bézier per rendere ϱ(DP) minimale. □

Il calcolo esplicito di DP richiede la differenziazione della mappa di Poincaré rispetto alle condizioni iniziali, effettuata tramite integrazione delle equazioni di variazione — le equazioni linearizzate lungo la traiettoria nominale. Questo calcolo, pur non disponendo di una forma chiusa in generale, è efficiente dal punto di vista numerico: ogni integrazione delle equazioni di variazione è un sistema lineare a coefficienti periodici, risolvibile con la matrice fondamentale di Floquet.

3.6 Ottimizzazione di Traiettoria con DDP e iLQR per Locomozione Bipede

Il MPC centroidale del Capitolo 2 pianifica le forze di contatto su orizzonti corti (0.5–1 s). Per la pianificazione su orizzonti più lunghi (2–5 passi) con ottimizzazione congiunta di postura e contatti, si adottano metodi di shooting differenziabile basati sul Differential Dynamic Programming (DDP) o sulla sua variante iterativa iLQR.

Differential Dynamic Programming: derivazione

Il DDP risolve il problema di ottimizzazione di traiettoria a orizzonte finito:

$$\min_{u_0,\dots,u_{N-1}} \Sigma_k\, l(x_k, u_k) + l_k(x_N)$$

$$\text{s.t. } x_{k+1} = f(x_k, u_k),\ x_0 \text{ dato}$$

dove $x = (\eta, \dot{\eta}) \in \mathbb{R}^{2(6+n)}$ è lo stato, $u = \tau$ le coppie, f la dinamica discreta del robot. Il DDP opera tramite il principio di ottimalità di Bellman: la

funzione valore $V(x_⊙) = \min_{u_⊙:⊙} \Sigma_{⊙≥⊙}\, l(x_⊙,u_⊙)$ soddisfa la ricorrenza $V(x_⊙) = \min_u [l(x,u) + V(f(x,u))]$. Il DDP linearizza questa ricorrenza attorno a una traiettoria nominale $(\bar{x}, \bar{u})$, espandendo al secondo ordine:

$$Q(x,u) = l(x,u) + V(f(x,u))$$

$$\approx Q_0 + Q_x^T \delta x + Q\_u^T \delta u + \tfrac{1}{2} \delta x^T Q\_\{xx\} \delta x + \delta x^T Q\_\{xu\} \delta u + \tfrac{1}{2} \delta u^T Q\_\{uu\} \delta u$$

dove i coefficienti $Q\_\{xx\}$, $Q\_\{xu\}$, $Q\_\{uu\}$ sono costruiti dalle derivate seconde di l e V, calcolate ricorsivamente dalla fine dell'orizzonte verso l'inizio. Il passo di controllo ottimale è $\delta u^* = -Q\_\{uu\}^{-1}(Q\_u + Q\_\{xu\}^T \delta x)$, che definisce il guadagno di feedback $K = Q\_\{uu\}^{-1}\, Q\_\{xu\}^T$ e il feedforward $k = Q\_\{uu\}^{-1}\, Q\_u$.

iLQR: la variante regolare

L'iLQR (iterative LQR, Todorov & Li 2004; Li & Todorov 2004) è la versione di DDP che neglinge le derivate seconde della dinamica $f\_\{xx\}$, $f\_\{xu\}$, $f\_\{uu\}$ nel calcolo dei $Q\_\{xx\}$ e $Q\_\{xu\}$. Questo corrisponde a una linearizzazione più grossolana, ma ha il vantaggio di richiedere solo le derivate prime $f\_x = \partial f/\partial x$ e $f\_u = \partial f/\partial u$, disponibili tramite l'algoritmo RNEA differenziato (§9.3). La convergenza dell'iLQR è superlineare per problemi con dinamica quasi-lineare, e il costo per iterazione è $O(N\, n^3)$ con N l'orizzonte e n la dimensione dello stato.

Per la locomozione bipede, la dinamica f include sia la fase di singolo appoggio (equazioni di Eulero-Lagrange) sia gli eventi di impatto (mappa di reset, §3.3.1). Il DDP/iLQR ibrido gestisce questi eventi tramite la differenziazione della mappa di Poincaré $DP(x^*) = \partial\Delta/\partial x^- \cdot \Phi(T; x^*)\,|\_Z$, già introdotta nel Teorema 3.2. Il framework FROST (Fast Robot Optimization and Simulation Toolkit, Hereid & Ames, 2017) implementa l'iLQR ibrido per sistemi bipedi e produce traiettorie ottimali per HZD che possono essere usate come riferimento per il WBC.

Proposizione 3.2 (Convergenza locale dell'iLQR per sistemi smooth).

Sia u^* la sequenza di controllo ottimale per il problema a orizzonte N. Se la dinamica f è C^2 e le matrici $Q\_\{uu\}(k)$ sono definite positive per ogni k, allora l'iLQR converge con tasso superlineare a u^* da condizioni iniziali

sufficientemente vicine, con un residuo della backward pass che scala come $O(\|f_{xx}\| \|\delta u\|^2)$ *per iterazione.*

Dimostrazione. La convergenza superlineare segue dall'analisi dello shooting a multipla stadi: il passo iLQR equivale a un Newton step per il problema di ottimizzazione di traiettoria, con un errore di approssimazione dovuto ai termini di ordine due della dinamica f_{xx}. Per f C^2 e step-size regolato tramite line search, la convergenza è assicurata. Il dettaglio tecnico è in Mayne (1966) e Tassa-Todorov (2014). □

Capitolo 3.7 — Raccordo tra HZD e Reinforcement Learning: Inizializzazione e Stabilizzazione

Raccordo con la monografia: Cap. 3.3–3.5 (HZD), Cap. 23–24 (RL fisico, curriculum learning).

7.1 La Connessione Fondamentale

I cicli limite HZD (Cap. 3.3–3.5) e le policy RL (Cap. 23) sono due modi di descrivere lo stesso fenomeno -- la locomozione periodica bipede -- da prospettive complementari. HZD fornisce un framework analitico rigoroso per la sintesi e l'analisi di cicli limite stabili, ma richiede la progettazione manuale degli output virtuali e dei coefficienti di Bezier. RL non richiede progettazione manuale ma soffre di esplorazione inefficiente e convergenza lenta per task di locomozione su terreni non strutturati.

La sinergia tra i due approcci si manifesta in tre modi: (1) i cicli limite HZD come inizializzazione della policy RL, (2) le zero dynamics come spazio ridotto per l'RL, (3) la mappa di Poincare come funzione di reward analitica.

7.2 Inizializzazione RL da Cicli Limite HZD

La policy di controllo HZD (u* = u*_HZD dalla sezione 3.5) fornisce un'inizializzazione eccellente per la policy RL perche garantisce un ciclo di camminata stabile su terreno piano. Il training RL inizia dalla policy HZD e la perfeziona per gestire perturbazioni e terreni non strutturati:

pi_0^RL = pi_HZD

pi_{k+1}^RL = argmax_pi E_{tau ~ pi} [R(tau)] s.t. D_KL(pi || pi_k^RL) <= delta

Il vincolo TRPO (eqn. in Cap. 23.2) garantisce che la policy non si allontani troppo dall'inizializzazione HZD nelle prime iterazioni, preservando la stabilita del ciclo di camminata. Il numero di campioni necessari per il training RL con inizializzazione HZD e tipicamente 10-30x inferiore rispetto all'inizializzazione casuale, grazie al fatto che la policy iniziale e gia vicina all'ottimo su terreno piano.

7.3 Le Zero Dynamics come Spazio Ridotto per l'RL

Le zero dynamics di HZD (sezione 3.3.3) riducono il sistema da 2n dimensioni a 2 (theta, theta_dot). Questa riduzione puo essere sfruttata per il training RL: invece di ottimizzare la policy nello spazio completo TQ, si ottimizza solo nel sottospazio delle zero dynamics, lasciando che il controller HZD gestisca i restanti n-1 gradi di liberta attuati.

Formalmente, la policy RL ridotta e pi_ZD: (theta, theta_dot) -> delta_alpha, dove delta_alpha e la perturbazione dei coefficienti di Bezier (sez. 3.3.2) rispetto alla traiettoria nominale HZD. Il reward di RL viene valutato sulla traiettoria

risultante, e il gradiente e calcolato rispetto a delta_alpha invece che ai parametri della policy completa.

Proposizione S7.1 (Riduzione della complessita campionaria con RL sulle zero dynamics).

Sia N_full il numero di campioni necessario per il training RL nello spazio TQ (dim 2n) e N_ZD quello per il training RL nello spazio delle zero dynamics (dim 2). Sotto l'ipotesi che le zero dynamics catturino i gradi di liberta rilevanti per la stabilita orbitale, il rapporto di complessita campionaria e N_ZD / N_full = O((2/2n)^gamma) = O(n^{-gamma}) dove gamma > 0 dipende dal grado di smoothness della funzione valore. Per n = 15 (un bipede con 15 DOF articolari) e gamma = 0.5, il risparmio e di circa 4x.

7.4 La Mappa di Poincare come Reward Analitica

Il raggio spettrale della mappa di Poincare rho(DP(x*)) (Teorema 3.1) e una misura diretta della stabilita orbitale del ciclo di camminata: rho < 1 garantisce la convergenza al ciclo, e rho piccolo indica convergenza rapida. Questa quantita puo essere usata come reward supplementare nel training RL, penalizzando le policy che producono cicli con rho vicino a 1 (marginalmente stabili):

R_Poincare(pi) = -lambda * max(0, rho(DP_{pi}(x*)) - rho_target)

dove rho_target e il raggio spettrale target (tipicamente 0.7-0.8) e lambda e il peso. Il calcolo di rho(DP) richiede l'integrazione delle equazioni di variazione lungo il ciclo nominale (sez. 3.3.4), effettuabile in O(n^2 T) operazioni per un ciclo di durata T. Questo reward analitica complementa il reward sparso di velocita, fornendo un segnale denso sulla qualita della stabilita orbitale.

Capitolo 4 — Contatto Unilaterale e Sistemi Lagrangiani Vincolati

Il contatto tra un robot e l'ambiente è un fenomeno di grande complessità che sfida le fondamenta della meccanica classica dei sistemi vincolati: il vincolo è unilaterale (il suolo può spingere ma non tirare), discontinuo (il contatto si apre e si chiude) e accoppiato all'attrito (la forza tangenziale è limitata dalla forza normale). La formulazione matematica rigorosa richiede strumenti che vanno dalla meccanica lagrangiana con vincoli non olonomi alla teoria delle inclusioni differenziali e delle disuguaglianze variazionali.

4.1 Vincoli Lagrangiani e il Sistema DAE

Con più vincoli Φ(q) = 0, Φ: Q → ℝ^k, le equazioni del moto diventano il sistema DAE:

$$M(q)\,\ddot{q} \;+\; C(q,\dot{q})\,\dot{q} \;+\; g(q) \;=\; \tau \;+\; J(q)^{\top}\lambda\,, \quad \Phi(q) \;=\; 0 \quad (4.1)$$

dove J(q) = ∂Φ/∂q ∈ ℝ^{k×n}. Differenziando Φ(q) = 0 due volte rispetto al tempo si ottiene la condizione di accelerazione J(q)q̈ + J̇(q)q̇ = 0, che produce il sistema lineare per la determinazione simultanea di q̈ e λ:

$$[M \quad -J^{\top}\,;\; J \;\; 0][\ddot{q}\,;\,\lambda] \;=\; [\tau \;-\; C\dot{q} \;-\; g\,;\quad -J\dot{q}] \quad (4.2)$$

La matrice di sistema è la matrice di Kuhn-Tucker del problema di ottimizzazione associato. La sua invertibilità richiede che M sia definita positiva (sempre vera per sistemi fisici) e che J abbia rango pieno per righe.

4.2 Condizioni di Signorini e Problema di Complementarità

Il contatto unilaterale introduce la struttura di complementarità che distingue il problema da quello dei vincoli bilaterali. Sia φ_n(q) la funzione di gap: le condizioni di Signorini sono:

$$\varphi_n(q) \geq 0, \quad \lambda_n \geq 0, \quad \varphi_n(q)\cdot\lambda_n \;=\; 0 \quad (4.3)$$

In presenza di attrito di Coulomb $\|\lambda_t\| \le \mu\, \lambda_n$, la formulazione più comune per simulazione e controllo è il Linear Complementarity Problem (LCP), ottenuto approssimando il cono di attrito con una piramide a d facce.

> **Teorema 4.1 (Esistenza della soluzione per il LCP di contatto).**
> *Sia $H(q) = J(q)\, M(q)^{-1}\, J(q)^\top \in \mathbb{R}^{k\times k}$ la matrice di Delassus. Se $H(q)$ è definita positiva — condizione garantita se $J(q)$ ha rango pieno per righe — allora per ogni vettore $q \in \mathbb{R}^k$ il LCP (H, q) ha soluzione unica, calcolabile tramite l'algoritmo di Lemke in un numero finito di passi.*

Dimostrazione. Poiché M(q) è simmetrica definita positiva e J(q) ha rango pieno per righe, H(q) è il prodotto congruente di una matrice SPD con una matrice di rango pieno, dunque è essa stessa SPD. Una matrice SPD è una P-matrice. Per il teorema di Murty e Cottle (1980), ogni LCP con P-matrice ha soluzione unica, e l'algoritmo di Lemke termina in un numero finito di passi per P-matrici. □

4.3 Legge di Impatto e Integratori Time-Stepping

La legge di impatto di Newton-Darboux con coefficiente di restituzione $e \in [0,1]$ modella la discontinuità delle velocità al momento del contatto:

$$\dot{q}^{+} = \dot{q}^{-} + M(q)^{-1} J(q)^{\top} \Lambda, \quad J(q)\,\dot{q}^{+} = -e \cdot J(q)\,\dot{q}^{-} \tag{4.4}$$

La soluzione per l'impulso Λ è $\Lambda = -(1+e)\, H(q)^{-1}\, J(q)\, q^{-}$. Gli integratori time-stepping, come quelli di Stewart-Trinkle, discretizzano direttamente le condizioni di complementarità nell'intervallo [t, t+h] risolvendo un LCP per ogni passo. Questo approccio è incondizionatamente stabile — nel senso che non richiede la riduzione del passo per garantire la stabilità — ed è la base algoritmica di simulatori come MuJoCo, DART e Bullet.

4.4 Il principio di minima costrizione di Gauss

Il principio di Gauss (1829), noto anche come principio della minima costrizione (Prinzip des kleinsten Zwanges), fornisce una caratterizzazione variazionale dell'accelerazione di un sistema soggetto a vincoli, del tutto equivalente alle equazioni di Lagrange ma formulata

come problema di ottimizzazione. La sua utilità nella robotica è triplice: offre un'interpretazione geometrica della proiezione del moto nel null-space dei vincoli, si presta direttamente alla formulazione QP del WBC, e generalizza in modo naturale al caso di vincoli unilaterali (disuguaglianze) tramite la variante di Gauss-Hertz.

Si consideri un sistema a n gradi di libertà con vettore di accelerazione libera q̈^{free} (l'accelerazione che si avrebbe in assenza di vincoli, data dalle forze generalizzate attive) e uno o più vincoli bilaterali $\Phi(q) = 0$ di Jacobiano $J(q) = \partial\Phi/\partial q \in \mathbb{R}^{k\times n}$. Il principio di Gauss afferma che l'accelerazione reale $\ddot{q}$ minimizza la funzione di costrizione di Gauss:

$$Z(\ddot{q}) = \tfrac{1}{2}\,(\ddot{q} - \ddot{q}^{free})^{\top}\, M(q)\,(\ddot{q} - \ddot{q}^{free})$$

soggetta al vincolo di compatibilità delle accelerazioni $J(q)\,\ddot{q} + \dot{J}(q)\,\dot{q} = 0$ (derivata seconda di $\Phi(q(t)) = 0$ rispetto al tempo). Geometricamente, la funzione di costrizione Z è la norma quadratica dell'accelerazione nella metrica M rispetto all'accelerazione libera: il sistema fisico sceglie l'accelerazione vincolata più vicina, nel senso della metrica inerziale, all'accelerazione non vincolata.

Il problema di minimizzazione vincolata si scrive esplicitamente come:

$$\min_{\ddot{q}} \tfrac{1}{2}\,\ddot{q}^{\top} M(q)\,\ddot{q} - \ddot{q}^{\top}\,(\tau - C(q,\dot{q})\dot{q} - g(q)) \quad s.t.\;\; J(q)\,\ddot{q} = -\dot{J}(q)\,\dot{q}$$

dove si è usato il fatto che $M(q)\,\ddot{q}^{free} = \tau - C\,\dot{q} - g$. Le condizioni di ottimalità KKT sono:

$$M(q)\,\ddot{q} - (\tau - C\dot{q} - g) = J(q)^{\top}\,\lambda$$

$$J(q)\,\ddot{q} + \dot{J}(q)\,\dot{q} = 0$$

che coincidono esattamente con il sistema DAE (4.2) della monografia. Il principio di Gauss è quindi equivalente alle equazioni di Lagrange: non introduce nuova fisica, ma fornisce una formulazione come QP vincolato che è computazionalmente vantaggiosa e spiega perché le forze di vincolo (i moltiplicatori di Lagrange λ) abbiano l'interpretazione di forze che minimizzano il lavoro della costrizione.

Corollario 4.1 (Proiezione ortogonale di Gauss). *L'accelerazione reale $\ddot{q}$ è la proiezione ortogonale — rispetto alla metrica M — dell'accelerazione libera $\ddot{q}^{free}$ sullo spazio affine delle accelerazioni compatibili con i vincoli $K = \{\ddot{q} : J\ddot{q} = -\dot{J}\dot{q}\}$.*

Dimostrazione. Il problema di minimizzazione di Z è un QP convesso in $\ddot{q}$ con matrice oggettivo $M > 0$. Lo spazio ammissibile K è un affine sottospazio (aperto) di $\mathbb{R}^n$. La soluzione è la proiezione ortogonale rispetto alla norma $\|\cdot\|_M$ su K, data dall'equazione di proiezione $M(\ddot{q} - \ddot{q}^{free}) \in \mathrm{col}(J^\top)$, ovvero $M(\ddot{q} - \ddot{q}^{free}) = J^\top \lambda$ per qualche λ, il che riproduce la prima condizione KKT.

□

La generalizzazione al caso unilaterale — contatto che può aprirsi e chiudersi — sostituisce il vincolo di uguaglianza $J\ddot{q} = -\dot{J}\dot{q}$ con il vincolo di complementarità:

$$\varphi_n(q) \geq 0, \quad \lambda_n \geq 0, \quad \varphi_n(q)\,\lambda_n = 0$$

Il problema variazionale corrispondente è una variante del principio di Gauss su un cono convesso anziché su un sottospazio lineare. La soluzione è il proiettore metrico-indotto di $\ddot{q}^{free}$ sul cono ammissibile $K_+ = \{\ddot{q} : J\ddot{q} \geq -\dot{J}\dot{q}\}$, un insieme convesso chiuso in $\mathbb{R}^n$. L'esistenza e l'unicità di questa proiezione — garantite per K_+ non vuoto e $M > 0$ — giustificano dal punto di vista variazionale l'esistenza della soluzione del LCP di contatto già enunciata nel Teorema 4.1.

4.4 Derivazione delle Condizioni di Signorini dal Principio del Minimo Potenziale Complementare

Le condizioni di Signorini — $\phi_n \geq 0$, $\lambda_n \geq 0$, $\phi_n \lambda_n = 0$ — non sono postulate ma si derivano come condizioni di ottimalità di un principio variazionale. Questo aggancia la teoria del contatto alla meccanica variazionale e fornisce il contesto corretto per la simulazione differenziabile.

Sia $\phi_n(q) \geq 0$ la funzione di gap normale al contatto (distanza con segno tra le superfici). Il **potenziale di contatto unilaterale** è la funzione indicatrice del cono non negativo:

$$\Psi(\phi_n) = \{0 \; se \; \phi_n \geq 0 \quad + \infty \; se \; \phi_n < 0$$

Il principio del minimo potenziale complementare per il sistema con contatti afferma che la forza di reazione normale λ_n è la soluzione del problema di minimizzazione:

$$\lambda_n^* = arg \min_{\lambda_n \geq 0} \left[\frac{1}{2} \lambda_n^\top H \lambda_n + q^\top \lambda_n + \Psi^*(\lambda_n) \right]$$

dove $\Psi^*(\lambda_n) = sup_{\phi_n} \left[\lambda_n^\top \phi_n - \Psi(\phi_n) \right]$ è la coniugata di Fenchel-Legendre di Ψ. Per la funzione indicatrice del cono non negativo, $\Psi^*(\lambda_n) = 0$ per $\lambda_n \geq 0$ e $\Psi^*(\lambda_n) = + \infty$ per $\lambda_n < 0$. Le condizioni di ottimalità del primo ordine (KKT) del problema di minimizzazione danno:

$$H\lambda_n^* + q \in - \partial \Psi^*(\lambda_n^*), \quad \lambda_n^* \geq 0$$

dove $\partial \Psi^*$ è il subgradiente. Poiché Ψ^* è la funzione indicatrice del cono non negativo, $- \partial \Psi^*(\lambda_n) = N_{R_{\geq 0}}(\lambda_n)$ è il cono normale al cono non negativo. Questo si traduce nella condizione:

$$\phi_n = H\lambda_n^* + q \geq 0, \quad \lambda_n^* \geq 0, \quad \phi_n^\top \lambda_n^* = 0$$

che è precisamente la condizione di Signorini nella forma del Problema di Complementarità Lineare (LCP). La derivazione mostra che le condizioni di Signorini sono equivalenti alle condizioni KKT di un problema di minimizzazione convessa, e che la forza di contatto è il moltiplicatore duale del vincolo di non-penetrazione.

Conseguenza. La formulazione del contatto come problema di minimizzazione convessa implica che la matrice H (matrice di Delassus) deve essere semidefinita positiva affinché il LCP abbia soluzione. Questo è

garantito dalla struttura $H = J_a M^{-1} J_a^\top$ con M definita positiva e J_a di rango pieno per righe.

4.5 Teorema di Esistenza e Unicità del LCP di Delassus

La domanda di quando il LCP del contatto ha soluzione unica ha risposta precisa in termini della struttura della matrice di Delassus.

Definizione 4.1 (Matrice copositive e P-matrice). Una matrice $H \in R^{k\times k}$ è: - **Copositiva** se $x^\top Hx \geq 0$ per ogni $x \geq 0$. - **Strettamente copositiva** se $x^\top Hx > 0$ per ogni $x \geq 0$, $x \neq 0$. - P**-matrice** se tutti i suoi minori principali sono positivi.

Teorema 4.2 (Cottle-Dantzig 1968, applicato al LCP di Delassus). *Sia $H = J_a M^{-1} J_a^\top$ la matrice di Delassus ristretta ai contatti attivi con $J_a \in R^{k\times(6+n)}$ di rango pieno per righe ($k \leq 6 + n$). Allora:*

(i) H è semidefinita positiva e il LCP(H, q) ha sempre soluzione.

(ii) Se H è definita positiva, il LCP(H, q) ha soluzione unica per ogni $q \in R^k$.

(iii) La soluzione unica λ^* è continua rispetto a (H, q) quando H è definita positiva.

Dimostrazione. (i) La semidefinitezza positiva di H segue da: per ogni $v \in R^k$, $v^\top Hv = v^\top J_a M^{-1} J_a^\top v = \|M^{-1/2} J_a^\top v\|^2 \geq 0$. L'esistenza della soluzione del LCP con H semidefinita positiva segue dal teorema di Lemke applicato a LCP con matrice cocopositiva (ogni matrice semidefinita positiva è cocopositiva).

(ii) Se J_a ha rango pieno per righe, allora per ogni $v \neq 0$, $J_a^\top v \neq 0$ (per il rango pieno), e quindi $v^\top Hv = \|M^{-1/2} J_a^\top v\|^2 > 0$ poiché $M^{-1/2}$ è invertibile. Quindi H è definita positiva. Per LCP con H definita positiva, la funzione di costo $f(\lambda) = \frac{1}{2}\lambda^\top H\lambda + q^\top \lambda$ è strettamente convessa su $R^k_{\geq 0}$, dunque il minimo è unico. La condizione KKT del

minimo è precisamente la condizione LCP, quindi la soluzione è unica.

(iii) La continuità di $\lambda^*(H, q)$ rispetto a (H, q) nella regione di H definita positive segue dalla continuità della proiezione su convex set e dal teorema della funzione implicita applicato alle condizioni KKT attive. ▫

Corollario 4.1 (Configurazioni singolari). *Il LCP di Delassus perde l'unicità della soluzione nelle configurazioni singolari dove J_a non ha rango pieno per righe: questo accade quando i k vettori normali ai siti di contatto sono linearmente dipendenti (ad esempio, tre contatti quasi-complanari sul piano di appoggio). In tali configurazioni, la matrice H è semidefinita positiva ma non definita positiva, e il LCP può avere infinite soluzioni o, in presenza di attrito, nessuna soluzione.*

4.6 Legge di Impatto di Stronge e Approfondimento degli Integratori Time-Stepping

Legge di Impatto Energeticamente Consistente

La legge di impatto di Newton ($\lambda_n^+ = -e\lambda_n^-$, con e coefficiente di restituzione) ha un difetto fondamentale: per sistemi multi-contatto con $k \geq 2$, può violare la conservazione dell'energia anche per $e < 1$, producendo soluzioni che *aggiungono* energia durante l'impatto. La legge di Stronge (1990) corregge questo difetto.

La **legge di impatto di Stronge** definisce il coefficiente di restituzione energetico e^* come:

$$e^* = \sqrt{-\frac{W_r}{W_c}}$$

dove $W_c = \int_0^{t_{peak}} \lambda_n \dot{\phi}_n \, dt$ è il lavoro compiuto dalle forze di contatto durante la fase di compressione (negativo, poiché $\dot{\phi}_n < 0$) e $W_r = \int_{t_{peak}}^{t_{sep}} \lambda_n \dot{\phi}_n \, dt$ è il lavoro durante la fase di restituzione (positivo). La condizione $e^* \in [0, 1]$ è

equivalente alla dissipazione di energia durante l'impatto. Per $e^* = 0$ si ha impatto perfettamente plastico, per $e^* = 1$ impatto perfettamente elastico.

L'implementazione numerica della legge di Stronge richiede l'integrazione del LCP in forma incrementale rispetto all'impulso di contatto $p_n = \int \lambda_n \, dt$, con la condizione di terminazione che l'impulso di restituzione sia $p_r = e^{*2} p_c$.

Integratore Time-Stepping di Stewart-Trinkle

L'integratore time-stepping di Stewart e Trinkle (1996) è il fondamento teorico di MuJoCo e della maggioranza dei simulatori fisici differenziabili. A differenza degli integratori event-driven (che individuano esattamente ogni evento di contatto), il time-stepping tratta i contatti come vincoli impulsivi in ogni passo temporale di larghezza h, senza rilevare esplicitamente gli istanti di impatto.

Il passo dell'integratore time-stepping risolve il seguente LCP misto (MLCP) in $\left(v_{t+h}, \lambda_h\right)$:

$$M\left(q_t\right)\left(v_{t+h} - v_t\right) = h\left[S\tau_t + J_c^{\top}\left(q_t\right)\lambda_h + f_{ext}\right]$$

$$\phi_n\left(q_t\right) + hJ_n\left(q_t\right)v_{t+h} \geq 0, \quad \lambda_{n,h} \geq 0$$

$$\left[\phi_n\left(q_t\right) + hJ_n\left(q_t\right)v_{t+h}\right]^{\top}\lambda_{n,h} = 0$$

con i vincoli di attrito di Coulomb approssimati da un cono poligonale. Questo MLCP ha soluzione unica se M è definita positiva e J_c ha rango pieno.

Teorema 4.3 (Consistenza e ordine dell'integratore time-stepping). *Sia $v^*(t)$ la soluzione esatta del sistema con contatti. L'integratore time-stepping di Stewart-Trinkle con passo h produce una soluzione $v_h(t)$ tale che:*

$$\| v_h(t) - v^*(t) \| = O(h)$$

nelle fasi di contatto persistente, e $O(1)$ negli istanti di impatto (discontinuità di v^*), con errore globale $O(h)$ su intervalli finiti contenenti un numero finito di impatti.

Dimostrazione (sketch). Nelle fasi di contatto persistente, il time-stepping è equivalente a un metodo di Eulero implicito applicato all'equazione differenziale convenzionale, con errore locale $O\left(h^2\right)$ per passo e errore globale $O(h)$. Negli istanti di impatto, la discontinuità di v^* è approssimata dall'impulso $\lambda_{n,h}/h$ che converge alla misura di Dirac dell'impulso nell'impatto esatto per $h \to 0$. La stima globale $O(h)$ segue dalla teoria delle equazioni differenziali con misure come termine forzante (Schatzman 1998). ▫

Analisi dell'Errore di Energia negli Integratori Time-Stepping

Una proprietà non ovvia degli integratori time-stepping è la **dissipazione artificiale di energia**: anche per $e = 1$ (impatto perfettamente elastico), il time-stepping introduce una dissipazione numerica dell'ordine $O\left(h\|v\|^2\right)$ per passo. Questo è in contrasto con i sistemi hamiltoniani, dove gli integratori simplettici conservano esattamente l'energia.

Per la simulazione di robot umanoidi in locomozione, questa dissipazione numerica non è necessariamente indesiderabile: modella la dissipazione fisica dell'impatto del piede sul terreno e mantiene il sistema numericamente stabile durante la fase di appoggio. Tuttavia, per applicazioni che richiedono la conservazione dell'energia a lungo termine (pianificazione di traiettorie su orizzonti lunghi), si devono adottare integratori variazionali (Marsden-West 2001) che preservano esattamente la struttura simplettoidale del sistema con contatti.

Proposizione 4.2 (Dissipazione artificiale del time-stepping). *Per un sistema con singolo contatto elastico ($e = 1$) e massa m, passo h, il time-stepping di Stewart-Trinkle produce una perdita di energia cinetica per passo di:*

$$\Delta E_{num} = \frac{1}{2} m \|v_{t+h} - v^*(t + h)\|^2 = O\left(h^2 \| v_t\|^2\right)$$

La perdita cumulativa su T/h passi è $O\left(h\|v\|^2 T\right)$, lineare nel tempo e proporzionale al passo.

Dimostrazione. La discrepanza $v_{t+h} - v^*(t+h)$ è $O\left(h\|v_t\|\right)$ per il Teorema 4.3. L'errore energetico è $\frac{1}{2}m\|\left(v_{t+h} - v^*(t+h)\right)\|^2 = O\left(h^2 \| v_t\|^2\right)$. Accumulando su T/h passi e usando la diseguaglianza triangolare si ottiene l'errore cumulativo $O\left(h\|v\|^2 T\right)$. ▫

4.7 Modelli di contatto regolarizzati, meccanica di Hertz e convergenza degli integratori

Il trattamento del contatto tramite LCP e integratori time-stepping ha il vantaggio della coerenza fisica — le condizioni di Signorini sono esattamente soddisfatte a ogni passo — ma introduce una non-differenziabilità nelle transizioni di contatto che complica gli algoritmi di ottimizzazione. Per la simulazione differenziabile (discussa nel Capitolo 9) e per certi algoritmi di controllo, è preferibile adottare modelli di contatto regolarizzati che approssimano le condizioni di complementarità con funzioni continue o liscie.

Il modello di Kelvin-Voigt sostituisce la complementarità dura con la forza di contatto:

$$f_n(q, \dot{q}) = \max(0,\, k_n\, \varphi_n(q) + d_n\, \dot{\varphi}_n(q, \dot{q})) \cdot (-n)$$

dove $k_n > 0$ è la rigidezza di penetrazione, $d_n > 0$ è lo smorzamento viscoso, e il $\max(0, \cdot)$ garantisce la condizione di non trazione. Questo modello è C^0 ma non C^1 in $\varphi_n = 0$; tuttavia la singolarità è soltanto agli ingressi, e i simulatori come MuJoCo usano varianti di questo modello per la stabilità numerica a passi di integrazione grandi.

Una descrizione più accurata della meccanica del contatto è fornita dal modello di Hertz, sviluppato nel 1882 per il contatto elastico tra sfere. Per due corpi in contatto con raggi di curvatura R_1 e R_2 e moduli di Young E_1, E_2 con coefficienti di Poisson ν_1, ν_2, il modulo di Hertz effettivo è:

$$E^* = \left[(1-\nu_1^2)/E_1 + (1-\nu_2^2)/E_2\right]^{-1}, \quad R^* = \left[1/R_1 + 1/R_2\right]^{-1}$$

La forza di contatto normale in funzione della penetrazione $\delta \geq 0$ è:

$$f_n = (4/3)\, E^* \sqrt{R^*}\, \delta^{3/2}$$

La dipendenza non lineare f_n $\propto$ δ^{3/2} è caratteristica del contatto elastico tridimensionale ed è distinta dalla linearità del modello di Kelvin-Voigt. Il modello di Hertz è conservativo (senza dissipazione di energia per contatti elastici), e la sua inclusione nei simulatori differenziabili richiede la differenziazione della funzione f_n(δ) = (4/3) E* √R* δ^{3/2}, che è C^∞ per δ > 0 ma solo C^{1/2} in δ = 0, richiedendo particolare attenzione nella gestione del touchdown.

La convergenza degli integratori time-stepping per sistemi con contatto è un risultato delicato che richiede strumenti dell'analisi non liscia. Per i metodi di Stewart-Trinkle (1996), che discretizzano le condizioni di complementarità nell'intervallo [t, t+h] e le risolvono come un LCP per ogni passo, il seguente risultato di convergenza è disponibile:

> **Teorema 4.4 (Convergenza dell'integratore time-stepping, Stewart 2000).** *Sia q(t) la soluzione del sistema meccanico con contatto unilaterale e attrito di Coulomb nel senso delle misure differenziali (formulazione di Moreau). Siano q_h(t) le soluzioni dell'integratore di Stewart-Trinkle con passo h > 0. Sotto ipotesi di regolarità sui dati (Lipschitzianità delle forze attive e attrito di Coulomb classico), la convergenza in misura è garantita:*
>
> $$sup\_\{t \in [0,T]\} \|q\_h(t) - q(t)\| \to 0 \quad per \quad h \to 0$$

La dimostrazione di questo teorema — che è uno dei risultati più importanti della meccanica computazionale non liscia — si basa sulla compattezza della sequenza di soluzioni approssimate (garantita dall'energia finita) e sulla convergenza debole dei moltiplicatori di Lagrange discreti agli impulsi di reazione nel senso delle misure di Stieltjes. Non si ha in generale convergenza con ordine esplicito, poiché le discontinuità di velocità agli impatti limitano la regolarità delle soluzioni.

Per la simulazione differenziabile, la non-differenziabilità negli eventi di contatto — che corrisponde alle superfici di transizione nel problema ibrido — è gestita nei simulatori moderni (Genesis, MuJoCo MJX) tramite la smoothed complementarity: le condizioni di Signorini φ_n ≥ 0, λ_n ≥ 0, φ_n λ_n = 0 sono sostituite da:

$$\varphi\_n \, \lambda\_n = \varepsilon, \quad \varphi\_n \geq 0, \ \lambda\_n \geq 0 \quad \textit{(path following per } \varepsilon \to 0\textit{)}$$

o equivalentemente dalla formulazione Fischer-Burmeister: $\psi\_\varepsilon(\varphi\_n, \lambda\_n) = \varphi\_n + \lambda\_n - \sqrt{(\varphi\_n^2 + \lambda\_n^2 + \varepsilon^2)} = 0$, che è C^∞ per $\varepsilon > 0$ e converge alla complementarità dura per $\varepsilon \to 0$. Questa regolarizzazione consente il calcolo del gradiente $\partial\lambda\_n/\partial\theta$ (dove θ sono i parametri del simulatore) tramite la differenziazione implicita delle condizioni di stazionarità, che è il metodo adottato per la simulazione differenziabile discussa nel Capitolo 9.

PARTE II

Vision-Language-Action Models Di Nuova Generazione

Capitolo 5 — Da RT-2 a OpenVLA: Architetture Vision-Language-Action

La ricerca sui modelli vision-language-action (VLA) ha conosciuto un'accelerazione straordinaria a partire dal 2023, quando il lavoro Robotic Transformer 2 (RT-2) di Google DeepMind ha dimostrato che un modello linguistico di grandi dimensioni, pre-addestrato su testo e immagini, può essere raffinato per produrre direttamente azioni robotiche senza richiedere rappresentazioni intermedie esplicitamente progettate. Questo cambio di paradigma ha spostato il baricentro della robotica manipolativa dall'ingegneria delle rappresentazioni verso il problema del raffinamento efficiente di modelli a miliardi di parametri.

RT-2 si fonda sull'architettura PaLM-E, un modello multimodale che integra un modello linguistico da 540 miliardi di parametri con un encoder visivo ViT-4B. La chiave concettuale è la tokenizzazione delle azioni: le coppie (posizione, orientazione, apertura del gripper) dell'organo terminale vengono discretizzate in un vocabolario finito di token, trattati alla stregua delle parole del modello linguistico. In questo modo, la produzione di un'azione diventa indistinguibile, dal punto di vista dell'architettura, dalla produzione di una risposta testuale: il transformer autoregressivo campiona sequenzialmente i token dell'azione dal softmax sull'intero vocabolario.

OpenVLA, presentato da Hejna et al. nel 2024, ha reso questa classe di modelli accessibile alla comunità estendendo l'approccio RT-2 al modello Llama 2 da 7 miliardi di parametri. La scelta di un modello di dimensioni moderate ha consentito di effettuare il fine-tuning su hardware consumer — una singola GPU A100 — mantenendo prestazioni competitive su un ampio spettro di compiti manipolativi. Il modello produce le azioni nel formato stringa 'posizione_x posizione_y posizione_z rot_r rot_p rot_y apertura', dove ogni valore è un intero da 0 a 255 corrispondente a un bin della quantizzazione uniforme dello spazio d'azione.

5.1 Proiezione degli Embedding Linguistici negli Spazi d'Azione

Il meccanismo attraverso cui un modello linguistico produce azioni robotiche merita una trattazione matematica approfondita. Sia V il vocabolario del modello e E: V → ℝ^d lo spazio degli embedding, dove d è la dimensione nascosta (d ≈ 4096 per modelli da 7B). Il transformer autoregressivo implementa:

$$p(a_t \mid a_{1:t-1},\, o) \;=\; softmax(W_{head} h_t) \in \Delta^{\,|V|-1} \quad (5.1)$$

dove h_t = Transformer(E(a_{1:t-1}), E_{vis}(o)) ∈ ℝ^d è l'hidden state all'istante t e W_{head} ∈ ℝ^{|V|×d} è la matrice della testa di linguaggio. Il sottoinsieme A ⊂ V dei token d'azione occupa una frazione minima del vocabolario totale. Durante l'inferenza, la distribuzione di softmax viene mascherata sui soli token d'azione, normalizzando sulla probabilità condizionale:

$$p(a_t \mid a_{1:t-1},\, o,\, a_t \in A) \;=\; \frac{p(a\_t \mid a\_\{1:t-1\}, o)}{\sum\_\{a \in A\}\, p(a \mid a\_\{1:t-1\}, o)} \quad (5.2)$$

La convergenza del fine-tuning è governata dalla stabilità del gradiente nel trunk del modello linguistico. Per modelli di grandi dimensioni con full fine-tuning, il rischio di catastrophic forgetting — perdita delle rappresentazioni pre-addestrate — è significativo. Tecniche come LoRA (Low-Rank Adaptation), che limita gli aggiornamenti dei pesi a sottospazi di rango basso ΔW = BA con B ∈ ℝ^{m×r}, A ∈ ℝ^{r×n}, r ≪ rank(W), mitigano questo problema a costo di una riduzione della capacità di adattamento.

Il Trasferimento Composizionale e la Generalizzazione Zero-Shot

Una delle proprietà più sorprendenti dei modelli VLA è la capacità di eseguire compiti descritti verbalmente che non compaiono esattamente nel training set robotico, combinando la conoscenza semantica acquisita durante il pre-addestramento con le abilità motorie apprese dal fine-tuning. La base teorica di questo fenomeno risiede nell'ipotesi di distribuzione composizionale: se il training linguistico ha appreso rappresentazioni separate per oggetti e relazioni spaziali, il modello può generalizzare a nuove combinazioni. Questo è un caso particolare del principio di modularità funzionale nei sistemi cognitivi, e la sua verifica empirica sui benchmark di manipolazione generalizzata (LIBERO,

RoboVerse) costituisce uno dei risultati più rilevanti della letteratura recente.

5.3 Il meccanismo di attenzione multi-testa: derivazione completa e proprietà di approssimazione

Un layer di attenzione scalata con prodotto scalare (Scaled Dot-Product Attention) prende in ingresso tre matrici: query Q ∈ ℝ^{n×d_k}, chiavi K ∈ ℝ^{m×d_k} e valori V ∈ ℝ^{m×d_v}, dove n è il numero di token di query, m il numero di token di contesto, d_k la dimensione delle chiavi e d_v quella dei valori. Il meccanismo calcola:

$$\mathit{Attention}(Q, K, V) = \mathit{softmax}(Q\,K^{\top} / \sqrt{d_k})\,V \quad \in \mathbb{R}^{\{n \times d_v\}}$$

La matrice di attenzione A = softmax(Q K^⊤ / √d_k) ∈ ℝ^{n×m} è una matrice stocastica per righe (ogni riga somma a uno), in cui l'elemento A_{ij} misura quanto il token di query i "presta attenzione" al token di contesto j. Il fattore di scala 1/√d_k compensa l'aumento della varianza del prodotto scalare con la dimensione: se Q e K hanno componenti con varianza unitaria, allora Q_i · K_j ha varianza d_k, e senza normalizzazione il softmax saturerebbe per d_k grande, producendo gradienti quasi nulli.

Per comprendere perché softmax in corrispondenza di d_k grande satura, si osserva che per un vettore x ∈ ℝ^m con elementi di ordine √d_k, la derivata del softmax è σ(x)(I − σ(x) 1^⊤), che tende a zero quando x_max − x_min ≫ 1. Dividendo per √d_k si riporta la scala dei logit a O(1) in modo che il gradiente del softmax rimanga significativo durante l'apprendimento.

Il meccanismo di attenzione multi-testa (Multi-Head Attention, MHA) parallelizza h teste di attenzione su sottospazi di dimensione ridotta:

$$\mathit{MHA}(X) = \mathit{Concat}(\mathit{head}_1, \ldots, \mathit{head}_h)\,W^{O}$$

$$\mathit{head}_i = \mathit{Attention}(X\,W^{Q}_i,\; X\,W^{K}_i,\; X\,W^{V}_i)$$

dove W^Q_i, W^K_i ∈ ℝ^{d×d_k}, W^V_i ∈ ℝ^{d×d_v} e W^O ∈ ℝ^{hd_v × d} sono matrici di proiezione apprendibili, con d_k = d_v = d/h.

La complessità computazionale dell'MHA per sequenze di lunghezza n è $O(n^2 d)$, dominata dal calcolo del prodotto esterno n×n delle matrici di attenzione. Questa dipendenza quadratica in n è la ragione per cui i modelli VLA adottano strategie di compressione dei token visivi (downsampling con ViT) e masked attention efficiente.

> **Proposizione 5.1 (Capacità di approssimazione dell'attenzione).** *Sia f: {0,1}^n → ℝ^d una funzione su sequenze binarie di lunghezza n. Un singolo layer di attenzione con n teste e dimensione d_k = O(log n) può approssimare f con errore arbitrariamente piccolo per n sufficientemente grande, purché le matrici di proiezione siano scelte appropriatamente.*
>
> *Dimostrazione.* La dimostrazione utilizza la costruzione di Yun et al. (2020), che mostra che l'attenzione con testa singola può implementare operazioni di selezione e copia: per ogni coppia (i, j) di posizioni, si costruiscono Q = e_i e K = e_j (vettori canonici) in modo che A_{ij} ≈ 1 e A_{ij'} ≈ 0 per j' ≠ j. La combinazione di più teste consente di selezionare contemporaneamente insiemi di posizioni diverse, implementando ogni mappa di ristrutturazione delle sequenze. La combinazione con i layer feed-forward point-wise fornisce la completezza della rappresentazione. □

Un aspetto cruciale per i modelli VLA è la struttura causale dell'attenzione (causal masking). Poiché la generazione delle azioni deve essere autoregressiva — il token t-esimo viene generato condizionalmente ai token 1, ..., t−1 e alle osservazioni — la matrice di attenzione è mascherata con la maschera causale $M \in \{0, -\infty\}^{n \times n}$ con $M_{ij} = -\infty$ per $j > i$. La maschera causale preserva la struttura autocontrastiva del softmax: la distribuzione di probabilità condizionale $p(a_t \mid a_{1:t-1}, o)$ è implementata esattamente dalla composizione dei layer di attenzione causale.

La codifica posizionale: struttura sinusoidale e rotary embedding

Il transformer è intrinsecamente invariante per permutazione: senza informazione posizionale, la risposta a una sequenza (x_1, ..., x_n) è

identica alla risposta a qualsiasi sua permutazione. Per sequenze linguistiche o robotiche, la posizione temporale dei token è invece fondamentale per codificare l'ordine delle azioni. La codifica posizionale sinusoidale, introdotta in Vaswani et al. (2017), aggiunge all'embedding del token di posizione t un vettore PE(t) ∈ ℝ^d con componenti:

$$PE_{(t, 2i)} = sin(t / 10000^{2i/d})$$

$$PE_{(t, 2i+1)} = cos(t / 10000^{2i/d})$$

per i = 0, 1, ..., d/2 − 1. La base geometrica di queste formule è che la proiezione PE(t)^⊤ PE(t+k) dipende soltanto dall'offset k, non dalla posizione assoluta t: il modello può apprendere relazioni posizionali relative. Formalmente, si verifica che PE(t)^⊤ PE(t+k) = Σ_i cos(k / 10000^{2i/d}), funzione di k soltanto.

La codifica posizionale rotazionale (Rotary Position Embedding, RoPE), proposta da Su et al. nel 2021 e adottata nella maggioranza dei modelli LLM di nuova generazione (LLaMA 2-3, Mistral, GPT-NeoX), adotta un approccio radicalmente diverso: invece di aggiungere un vettore posizionale agli embedding, moltiplica i vettori di query e chiave per una matrice di rotazione dipendente dalla posizione. Per un vettore x ∈ ℝ^d e posizione t, RoPE applica:

$$x^{(RoPE)}_{(t)} = R_t\, x$$

dove R_t ∈ SO(d) è una matrice di rotazione a blocchi 2×2 della forma:

$$R_t = block\text{-}diag(rot(t\,\theta_1), rot(t\,\theta_2), ..., rot(t\,\theta_{d/2}))$$

con rot(α) = [[cos α, −sin α]; [sin α, cos α]] e θ_i = 10000^{−2(i−1)/d}. La proprietà fondamentale di RoPE è che il prodotto scalare tra query e chiave dopo la rotazione dipende soltanto dalla differenza di posizione:

$$(R_t\, q)^\top (R_s\, k) = q^\top R_{s-t}\, k = g(q, k, s-t)$$

dove la seconda uguaglianza segue dall'ortogonalità di R_t (R_t^⊤ R_s = R_{s−t} per la struttura a blocchi additivi degli angoli). Questa proprietà di equivarianza rispetto alla traslazione temporale è particolarmente preziosa nei modelli VLA: le sequenze di azioni robotiche hanno struttura

stazionaria (la relazione tra azioni consecutive dipende tipicamente dall'offset temporale, non dall'istante assoluto), e RoPE la codifica in modo intrinseco nell'architettura.

5.4 Fine-tuning dei modelli VLA: LoRA, catastrophic forgetting e RLHF robotico

Il problema del fine-tuning di un modello linguistico pre-addestrato su dati robotici è più delicato di quanto appaia: il modello base ha appreso rappresentazioni distribuite su miliardi di parametri che codificano la struttura del linguaggio naturale e la semantica visiva; un aggiornamento aggressivo di tutti i pesi per adattarsi alle azioni robotiche rischia di cancellare irrimediabilmente queste rappresentazioni — il fenomeno del catastrophic forgetting (dimenticanza catastrofica). La soluzione standard è il Low-Rank Adaptation (LoRA), ma le sue garanzie teoriche meritano un esame approfondito.

Sia $W \in \mathbb{R}^{m\times n}$ una matrice di pesi di un layer del transformer (ad esempio la matrice di proiezione delle query $W^Q \in \mathbb{R}^{d\times d}$). L'aggiornamento LoRA aggiunge una perturbazione di basso rango:

$$W' = W + \Delta W = W + \alpha/r \cdot B\,A, \quad \text{con } B \in \mathbb{R}^{m\times r},\ A \in \mathbb{R}^{r\times n},\ r \ll \min(m,n)$$

dove r è il rango dell'adattamento, α è il fattore di scaling e BA è inizialmente zero (A è inizializzata casualmente e B a zero). Il numero di parametri addizionali è r(m + n) per layer, invece di mn: per d = 4096 e r = 8, il risparmio è di un fattore 4096/(2×8) = 256 per layer. Per il fine-tuning su azioni robotiche, tipicamente si applicano gli adapter LoRA alle matrici di query e valore W^Q, W^V del transformer, mantenendo W^K, W^O e i pesi del feed-forward congelati.

> **Proposizione 5.2 (Approssimazione LoRA e distanza dai pesi originali).** *Sia W il peso originale con rango effettivo $r_{eff} = \text{rank}(W)$. Un aggiornamento LoRA $\Delta W = BA$ con rango $r < r_{eff}$ produce una variazione nella norma di Frobenius $\|\Delta W\|_F \le \|B\|_F\ \|A\|_F$. Per pesi inizializzati con schema He, $\|B\|_F =$*

$O(\sqrt{(mr)})$ e $\|A\|_F = O(\sqrt{(nr)})$, quindi $\|\Delta W\|_F = O(r\sqrt{(mn)})$, crescendo come $O(\sqrt{r})$ rispetto al rango per m,n fissati.

Dimostrazione. La disuguaglianza $\|\Delta W\|_F = \|BA\|_F \leq \|B\|_F \|A\|_F$ segue dall'inequazione di Cauchy-Schwarz per norme di Frobenius (sub-moltiplicatività). La stima su $\|B\|_F$ e $\|A\|_F$ con inizializzazione He gaussiana segue dalla concentrazione della norma di Frobenius di una matrice gaussiana m×r: $\|B\|_F \approx \sqrt{(mr)}$ con alta probabilità. □

La conseguenza pratica di questo risultato è che la norma $\|\Delta W\|_F/\|W\|_F$ rimane piccola per $r \ll r_{eff}$: la perturbazione è localmente piccola rispetto ai pesi originali, il che spiega l'efficacia di LoRA nel mitigare il catastrophic forgetting. Tuttavia, la norma spettrale $\|\Delta W\|_2 = \|BA\|_2 \leq \|B\|_2 \|A\|_2$ può essere non trascurabile anche per r piccolo, influenzando le rappresentazioni nei layer dove il segnale è dominato dalle componenti di rango basso.

Il Reinforcement Learning dal feedback umano (RLHF) è stato adottato in OpenVLA e nelle varianti di RT-2 per allineare la policy robotica con le preferenze umane su sicurezza e naturalezza del movimento. L'algoritmo Direct Preference Optimization (DPO, Rafailov et al. 2023) è diventato lo standard preferito rispetto al classico RLHF con reward model separato, per la sua stabilità e semplicità implementativa. Data una coppia di traiettorie (τ^+, τ^-) con τ^+ preferita da un supervisore umano rispetto a τ^-, la loss DPO è:

$$L_{DPO}(\theta) = -\log \sigma(\beta\,[\log(\pi_\theta(\tau^+)/\pi_{ref}(\tau^+)) - \log(\pi_\theta(\tau^-)/\pi_{ref}(\tau^-))])$$

dove π_{ref} è la policy di riferimento (il modello pre-addestrato congelato), β è un parametro di temperatura e σ è la funzione logistica. La derivazione di questa loss dal framework del reward model implicito (Bradley-Terry) mostra che minimizzare L_{DPO} equivale ad aumentare la probabilità relativa delle traiettorie preferite rispetto alla policy di riferimento, senza bisogno di un reward model esplicito. Il vantaggio in termini di stabilità rispetto al PPO (Proximal Policy Optimization) è significativo: L_{DPO} è una loss di classificazione convessa rispetto ai

logit log(π_θ(τ^+)/π_{ref}(τ^+)), mentre PPO richiede l'equilibrio delicato tra policy gradient e clip del rapporto.

5.5 Gemini Robotics e i Modelli VLA di Nuova Generazione (2025)

La pubblicazione di Gemini Robotics (Google DeepMind, 2025) ha segnato una discontinuità significativa nella linea RT-2 / OpenVLA: invece di raffinare un modello linguistico esistente su dati robotici, Gemini Robotics integra il modello multimodale Gemini 2.0 con un sistema di controllo a doppio livello che separa esplicitamente il ragionamento semantico a lungo orizzonte dal controllo motorio a bassa latenza. Questa separazione — denominata dual-process architecture — rispecchia la distinzione biologica tra il sistema prefrontale (pianificazione) e il sistema motorio-cerebellare (esecuzione).

Architettura a doppio processo

Il livello superiore (Gemini Planner) opera a 5–10 Hz e produce piani in linguaggio naturale o codice Python che descrivono la strategia di alto livello: "afferra il bicchiere con la mano destra applicando una presa pinch-grasp, porta al tavolo, ruota di 90° prima di rilasciare". Il livello inferiore (Motor Executor) opera a 50–100 Hz e traduce ogni istruzione del piano in una sequenza di pose di end-effector tramite un modello VLA a 3B parametri addestrato su dati di teleoperazione. Il trasferimento di informazioni tra i due livelli avviene tramite un linguaggio di interfaccia strutturato — chiamato Robot Action Language (RAL) — che specifica il tipo di primitiva, i parametri continui e le condizioni di terminazione.

Formalmente, sia $\pi_H: O \times G \to \Pi$ il pianificatore di alto livello (High-Level Planner) che mappa l'osservazione o e l'obiettivo g in un piano $\pi = (p_1, ..., p_◎)$ dove ogni $p_◎ \in \Pi$ è una primitiva RAL. Il livello basso $\pi_L: p_◎ \times O_{low} \to A$ è il controller che, data la primitiva corrente e le osservazioni ad alta frequenza, produce azioni continue. La struttura è formalmente analoga alla pianificazione TAMP (Capitolo 19) con la differenza che π_H è implementato da un LLM e le primitive π_L sono policy neurali invece di skill geometriche.

Gemini Robotics-ER: ragionamento spaziale avanzato

La variante Gemini Robotics-ER (Embodied Reasoning) introduce una capacità di ragionamento spaziale esplicito che mancava nei modelli RT-2 e OpenVLA: il modello è addestrato a produrre una stima 3D della scena (bounding box + orientazioni degli oggetti in SE(3)) come output ausiliario, prima di generare le azioni. Questo ragionamento spaziale esplicito — denominato spatial chain-of-thought — migliora significativamente le prestazioni su task che richiedono consapevolezza geometrica precisa, come l'inserimento di componenti, la manipolazione di oggetti orientati e l'esecuzione in ambienti cluttered.

Il meccanismo di attenzione spaziale è implementato tramite una testa di decodifica aggiuntiva che produce, per ogni oggetto rilevante nella scena, un token <obj_i> = (c_x, c_y, c_z, r, p, y, l, w, h) dove (c_x, c_y, c_z) è il centro del bounding box 3D, (r, p, y) sono gli angoli di Eulero e (l, w, h) le dimensioni. Questi token sono processati insieme ai token linguistici tramite cross-attention prima della generazione delle azioni, fornendo al modello un accesso esplicito alla geometria della scena in formato strutturato.

Risultati benchmark: Gemini Robotics raggiunge il 90.7% di successo su SimpleEval (task manipolativi semplici) e il 74.6% su ComplexEval (task con più step e oggetti nuovi), rispetto al 67% di π_0 e al 60% di OpenVLA su task equivalenti, secondo i dati riportati in Google DeepMind (2025).

5.6 Quantizzazione, Distillazione e Deployment su Hardware Embedded

I modelli VLA da 7B–55B parametri non sono direttamente deployabili su hardware robotico embedded per ragioni energetiche e di latenza: un forward pass di LLaMA-3 7B richiede ~2.4×10^{10} FLOP per token su sequenze di lunghezza 768 (immagine + testo), corrispondenti a ~120 ms su un SoC ARM Cortex-A78 a 3 TOPS — troppo lento per il controllo a 10 Hz. La riduzione della dimensione del modello tramite quantizzazione e distillazione è necessaria per il deploy su robot mobili con budget energetico di 10–50 W.

Quantizzazione a bassa precisione: analisi teorica

La quantizzazione dei pesi di un layer lineare $W \in \mathbb{R}^{m \times n}$ a b bit utilizza la trasformazione $Q_b(W) = \text{round}(W / \Delta) * \Delta$ dove $\Delta = (W_{max} - W_{min}) / (2^b - 1)$ è il passo di quantizzazione. L'errore di quantizzazione ha distribuzione approssimativamente uniforme su $[-\Delta/2, \Delta/2]$ con varianza $E[\varepsilon^2] = \Delta^2/12$. Per un layer con output $y = Wx$, la varianza dell'errore di output è:

$$\text{Var}(y_{err}) = \|x\|^2 \cdot n \cdot \Delta^2/12 = \|x\|^2 \cdot n \cdot (W_{max} - W_{min})^2 / (12 \cdot (2^b - 1)^2)$$

che decade come $O(4^{-b})$ al crescere dei bit. Per b = 4 rispetto a b = 8, l'errore di quantizzazione aumenta di un fattore 256, il che richiede l'applicazione di tecniche di calibrazione (SmoothQuant, GPTQ) per ridurre la varianza dei pesi nei layer critici.

GPTQ e OBQ: quantizzazione post-training ottimale

GPTQ (Frantar et al., 2022) è il metodo di quantizzazione post-training più adottato per modelli VLA. L'idea chiave è la quantizzazione ottimale riga per riga: per ogni riga $w \in \mathbb{R}^n$ della matrice di pesi, si trova la quantizzazione $\tilde{q}$ che minimizza il degradamento dell'output su un piccolo calibration set $X \in \mathbb{R}^{n \times m}$:

$$\tilde{q} = \text{argmin}_q \|(w - q)^T X\|^2 = \text{argmin}_q (w - q)^T H (w - q)$$

dove $H = XX^T/m$ è la matrice di Fisher empirica (Hessiana del quadratic loss). Il metodo GPTQ risolve questo problema tramite la variante Optimal Brain Quantization (OBQ) in $O(n^2)$ operazioni per riga, sfruttando la struttura sparsa di H per i modelli transformer. Il costo computazionale totale per quantizzare un modello da 7B parametri a 4 bit è di circa 30–60 minuti su un singolo nodo GPU A100.

Distillazione conoscenza per VLA compatti

La distillazione della conoscenza (Knowledge Distillation, KD) addestra un modello compatto "student" π_S a imitare il comportamento del modello completo "teacher" π_T. Per i modelli VLA, la loss di distillazione combina la loss behavioural (imitazione delle azioni) con la loss di feature alignment sugli hidden states:

$$L_{KD}(\theta_S) = \alpha\, L_{task}(\pi_S, \pi_T) + (1-\alpha) \sum_{l} \| h_{l,S}(o) - W_l\, h_{l,T}(o) \|^2$$

dove $h_{l,S}$ e $h_{l,T}$ sono gli hidden states del layer l nello student e nel teacher, $W_l \in \mathbb{R}^{d_S \times d_T}$ è una matrice di proiezione apprendibile, e α bilancia i due obiettivi. Il framework MobileVLA (2024) applica KD per distillare OpenVLA 7B in un modello da 300M parametri deployabile su Jetson AGX Orin (275 TOPS), ottenendo il 78% del successo del teacher su task standard con latenza di 50 ms.

Capitolo 6 — π_0 e le Architetture Cross-Embodiment

Il sistema π_0, sviluppato da Physical Intelligence, rappresenta un passo ulteriore nella generalizzazione dei modelli VLA: invece di addestrare un modello specializzato per un singolo robot, π_0 impara una policy condivisa su una varietà di piattaforme eterogenee, sfruttando la struttura comune dello spazio delle azioni e delle osservazioni per trasferire conoscenza tra robot con morfologie, attuatori e configurazioni sensoriali differenti. La versione π_0.5, pubblicata nel 2025, estende ulteriormente questa capacità a contesti domestici aperti.

L'architettura di π_0 combina un modello visio-linguistico pre-addestrato (basato su PaliGemma, 3B parametri) con un flow matching transformer per la generazione delle azioni. A differenza di RT-2, che tratta le azioni come token discreti, π_0 mantiene la natura continua dello spazio d'azione: la generazione avviene come un processo di denoising continuo, matematicamente equivalente all'integrazione di un'equazione differenziale ordinaria guidata da un campo vettoriale appreso.

6.1 Cross-Embodiment tramite Spazio d'Azione Comune

Il problema fondamentale del cross-embodiment è che robot diversi hanno spazi d'azione intrinsecamente incompatibili. La soluzione adottata in π_0 e nelle architetture RoboVLMs è la proiezione in uno spazio d'azione comune (Common Action Space, CAS) tramite encoding specifico per la morfologia. Sia $A_i \subseteq \mathbb{R}^{d_i}$ lo spazio d'azione del robot i-esimo, con d_i variabile. Si introducono per ogni robot una coppia di reti $\varphi_i: A_i \to \mathbb{R}^{d_a}$ e $\psi_i: \mathbb{R}^{d_a} \to A_i$, dove d_a è la dimensione del CAS, addestrate congiuntamente con il modello condiviso.

La giustificazione teorica risiede nell'ipotesi di invarianza della struttura compito: la distribuzione sulle traiettorie ottimali nel CAS è approssimativamente indipendente dalla morfologia del robot, nel senso che compiti semanticamente simili (afferrare un oggetto, aprire un cassetto) hanno strutture latenti simili indipendentemente dall'embodiment. Questa ipotesi è supportata empiricamente dai risultati

di π_0 e confermata teoricamente dalla teoria della rappresentazione invariante per le reti equivarianti.

6.2 Action Chunking come Modello Autoregressivo su Sequenze di Torsori

L'action chunking è la tecnica, introdotta sistematicamente nell'architettura ACT (Action Chunking with Transformers), per cui il modello predice non la singola azione al passo corrente ma un blocco di H azioni future consecutive, denominato chunk. Nel dominio SE(3), sia T_t $\in$ SE(3) la posa dell'organo terminale al tempo t e A_t = (T_t, T_{t+1}, …, T_{t+H-1}) $\in$ SE(3)^H il chunk di H pose future. Il modello predice la distribuzione condizionale p(A_t | o_{1:t}) sulla varietà SE(3)^H.

Poiché SE(3)^H non è uno spazio vettoriale, le azioni vengono parametrizzate come incrementi (twist) in se(3): ξ_k = Log_{SE(3)}(T⁻¹_{t+k} · T_{t+k+1}) $\in \mathbb{R}^6$, dove Log_{SE(3)} è il logaritmo di gruppo. Poiché se(3) $\cong \mathbb{R}^6$ è uno spazio vettoriale, la distribuzione su A_t può essere parametrizzata come una distribuzione gaussiana su $\mathbb{R}$^{6H}. Il modello ACT implementa la marginale tramite un Conditional Variational Autoencoder (CVAE):

$$L_{ACT}(\theta, \varphi) = E_{q\_\varphi}[log\ p\_\theta(A\_t \mid z,\ o\_\{1:t\})] - KL(q\_\varphi(z|A\_t,\ o\_\{1:t\}) \parallel p\_\theta(z|o\_\{1:t\})) \quad (6.1)$$

Il termine di ricostruzione è calcolato come log-verosimiglianza gaussiana sullo spazio tangente se(3)^H, usando la metrica riemanniana naturale su SE(3). Il termine KL ha una forma chiusa quando sia q_φ che p_θ sono distribuzioni gaussiane diagonali su $\mathbb{R}$^{d_z}: KL(N(μ_q, Σ_q) // N(μ_p, Σ_p)) = ½ [tr(Σ_p⁻¹ Σ_q) + (μ_p − μ_q)⊤ Σ_p⁻¹(μ_p − μ_q) − d_z + log(det Σ_p / det Σ_q)].

6.3 Equivarianza, invarianza e generalizzazione cross-embodiment

Un robot con n gradi di libertà articolari possiede un gruppo di simmetria cinematica G che agisce sullo spazio delle configurazioni Q: azioni g $\in$ G

producono configurazioni equivalenti. Per i manipolatori seriali, G include in genere le simmetrie di riflessione (un robot sinistro-destro simmetrico ha $G \cong \mathbb{Z}/2\mathbb{Z}$ sulla coordinata di lateralità), le simmetrie di rotazione planare per robot con struttura a colonna, e le simmetrie di traslazione lungo assi di ridondanza. Una policy $\pi: O \to A$ è equivariante rispetto a G se:

$$\pi(\rho\_O(g)\, o) = \rho\_A(g)\, \pi(o) \quad \textit{per ogni } g \in G,\, o \in O$$

dove ϱ_O e ϱ_A sono le rappresentazioni di G rispettivamente nello spazio delle osservazioni e in quello delle azioni. L'equivarianza implica che il comportamento della policy è coerente con le simmetrie del problema: se si specchia la scena visiva, la policy produce l'azione specchiata. L'invarianza è il caso speciale ϱ_A(g) = Id per ogni g.

Nella pratica del cross-embodiment, il gruppo di simmetria rilevante non è discreto ma continuo: il gruppo SE(3) agisce sulle pose dell'end-effector, e le azioni ottimali covarianti rispetto a SE(3) (ad esempio, afferrare un oggetto ruotato equivale ad applicare la stessa rotazione all'azione). Un modello VLA che rispetta questa equivarianza — denominato SE(3)-equivariante — ha per costruzione una capacità di generalizzazione zero-shot a pose dell'oggetto arbitrarie, senza dover campionare esplicitamente l'intero gruppo durante il training.

Il paradigma CTDE si fonda su una decomposizione fondamentale della funzione valore congiunta in contributi locali verificabili. La condizione IGM (Individual-Global-Max, Son et al. 2019) formalizza questa decomposizione: una famiglia di funzioni valore locali $\{Q^i\}_{i=1}^N$ soddisfa IGM rispetto al valore centralizzato Q_tot se $\text{argmax\_a } Q\_tot(s, a^1,...,a^N) = (\text{argmax\_}\{a^1\} Q^1(\tau^1, a^1),...,\text{argmax\_}\{a^N\} Q^N(\tau^N, a^N))$, dove τ^i è la local observation-action history dell'agente i. La condizione IGM garantisce che l'ottimizzazione decentralizzata (ogni agente sceglie l'azione che massimizza il proprio Q^i locale) produca la stessa azione congiunta dell'ottimizzazione centralizzata (max su tutto lo spazio congiunto).

QMIX (Rashid et al. 2018) implementa la condizione IGM attraverso una rete di mixing monotona: $Q\_tot(s, a^1,...,a^N; \theta\_mix) = g(s, Q^1(\tau^1,a^1;\theta^1),...,Q^N(\tau^N,a^N;\theta^N))$ dove g è una rete neurale con pesi non negativi

(garantiti attraverso una trasformazione softplus), che implementa una forma generalizzata di somma pesata non lineare. La non-negatività dei pesi di g garantisce la monotonia $\partial Q\_tot/\partial Q^i \geq 0$, che è condizione sufficiente per IGM. QMIX può quindi rappresentare l'intera classe delle funzioni valore congiunte monotone rispetto ai valori locali, includendo come caso speciale la somma semplice (VDN, Value Decomposition Networks) e come limite superiore la classe delle funzioni non fattorizzabili (Full-QMIX con rete mixing arbitraria).

Analisi teorica della classe espressiva di QMIX e MAPPO

> Proposizione 26.2 (Limitazione espressiva di QMIX). Esistono giochi cooperativi per cui il valore Q_tot ottimo non è rappresentabile come funzione monotona dei valori locali Q^i. Specificamente, qualsiasi gioco con struttura payoff non monotonicamente decomponibile richiede Q_tot non-QMIX-representable.
>
> Dimostrazione. Consideriamo un gioco a 2 agenti con matrice di payoff: $R(a^1=0, a^2=0) = 8$, $R(a^1=0, a^2=1) = -12$, $R(a^1=1, a^2=0) = -12$, $R(a^1=1, a^2=1) = 6$. Il massimizzatore congiunto è (0,0) con R=8. Ma per ogni funzione monotona $Q\_tot = g(Q^1, Q^2)$ con $\partial g/\partial Q^1, \partial g/\partial Q^2 \geq 0$, $\text{argmax}\_\{a^1\}\ Q^1$ e $\text{argmax}\_\{a^2\}\ Q^2$ devono concordare con $\text{argmax}\_\{a^1,a^2\}\ Q\_tot$. Se $Q\_tot(0,0) > Q\_tot(1,0)$ (per giustificare $a^1=0$), allora da $Q\_tot = g(Q^1, Q^2)$ con g monotona segue $Q^1(0) > Q^1(1)$. Analogamente $Q^2(0) > Q^2(1)$. Ma allora $\text{argmax}\_\{a^1,a^2\}\ Q\_tot = (0,0)$ indipendentemente da Q_tot(0,1) e Q_tot(1,1). Questo esclude la rappresentabilità di giochi con equilibri di Nash non dominating. □

MAPPO (Multi-Agent PPO) supera questa limitazione usando un critic centralizzato $V\_c(s^1,...,s^N)$ che non è soggetto alla condizione IGM: il critic stima il valore dello stato globale senza la necessità di fattorizzarsi in contributi locali. Il prezzo è che la decentralizzazione avviene solo a livello di policy (ogni π^i osserva solo o^i), mentre il critic centralizzato richiede accesso globale durante il training. Il criterio di convergenza per MAPPO nell'ambiente non stazionario (dove la distribuzione di stati cambia con il training delle policy degli altri agenti) è più delicato rispetto

al caso mono-agente: la condizione di Lyapunov per la stabilità asintotica richiede che il tasso di aggiornamento del critic sia sufficientemente più rapido del tasso di aggiornamento delle policy — la separazione delle scale temporali (two-timescale convergence) caratteristica degli algoritmi actor-critic.

La convergenza di MAPPO con N agenti segue dal seguente argomento di two-timescale stochastic approximation. Sia $\theta = (\theta^1,...,\theta^N)$ il profilo delle policy e φ i pesi del critic centralizzato. Gli aggiornamenti seguono: $\varphi_{t+1} = \varphi_t + \alpha_\varphi(t) \cdot \delta_\varphi(\theta_t, \varphi_t)$ e $\theta_i{}_{t+1} = \theta_i{}_t + \alpha_\theta(t) \cdot \delta_{\theta_i}(\theta_t, \varphi_t)$, dove $\alpha_\varphi(t) \gg \alpha_\theta(t)$ (critic veloce, policy lenta). Con questa separazione, il critic converge al suo punto fisso $\varphi^*(\theta)$ per ogni θ fisso (timescale fast), e le policy convergono poi al punto fisso θ^* del sistema dinamico lento. La condizione di punto fisso per le policy è esattamente l'equilibrio di Nash: nessun agente può incrementare il proprio reward variando unilateralmente la propria policy, dati i profili degli altri e il critic ottimale.

Il problema della non-stazionarietà è la principale sfida teorica del MARL rispetto al RL mono-agente. Dal punto di vista di un singolo agente i, il reward $R^i(s, a^i, a_{-i})$ dipende dalle azioni degli altri agenti a_{-i} che cambiano durante il training: l'ambiente effettivo che l'agente i percepisce è non-stazionario (si evolve con le policy degli altri agenti). Formalmente, la distribuzione marginale delle transizioni viste dall'agente i è $P^i(s'|s, a^i) = \Sigma_{a_{-i}} \Pi_{j\neq i} \pi^j(a^j|s^j) P(s'|s,a^i,a_{-i})$, che dipende dalle policy $\{\pi^j\}_{j\neq i}$ e quindi cambia durante il training. Questo viola l'ipotesi di stazionarietà del processo di Markov su cui si basa la convergenza dei metodi di TD standard, richiedendo l'uso di metodi robusti alla non-stazionarietà come il gradient clipping (PPO) o la trust region (TRPO).

Applicazioni robotiche: coordinamento di bracci multipli e flotte di umanoidi

Nei sistemi di assemblaggio automatizzato con N bracci robotici che condividono uno spazio di lavoro, la struttura del gioco cooperative MARL è quella di un potential game (§26.1) se il task è decomponibile in sotto-compiti indipendenti per braccio, ma diventa un gioco con esternalità negative in presenza di collisioni o conflitti di accesso a

componenti condivisi. La modellazione tramite CTDE permette di addestrare policy cooperative che evitano collisioni senza richiedere comunicazione esplicita a runtime: il critic centralizzato apprende implicitamente le politiche degli altri agenti durante il training, e la policy decentralizzata di ogni braccio produce comportamenti coordinati autonomamente. L'efficacia di questo approccio è stata dimostrata in sistemi di pick-and-place bimanuale con N=2 bracci Franka Panda (Zhao et al. 2023, §12), e i risultati scalano a $N \leq 8$ bracci con degradazione graceful della performance collettiva.

La costruzione di modelli equivarianti per SE(3) è un campo attivo. Un approccio è quello delle reti equivarianti basate sugli armonici sferici (Spherical CNNs, Cohen et al. 2018), che costruiscono feature map che si trasformano come rappresentazioni irriducibili del gruppo SO(3) sotto rotazione dell'input. Per il cross-embodiment, il vantaggio dell'equivarianza è quantificabile tramite il seguente risultato di generalizzazione:

> **Proposizione 6.1 (Riduzione della complessità campionaria per policy equivarianti).** *Sia π_G una policy G-equivariante addestrata su n_{tr} traiettorie di training campionate uniformemente su G-orbite di O. Il rischio empirico di una policy G-equivariante su un task con simmetria G soddisfa, con probabilità almeno $1-\delta$: $R(\pi_G) \leq \hat{R}(\pi_G) + O(\sqrt{(\log(1/\delta)/n_{tr}))}$, indipendentemente dall'ordine $|G|$ del gruppo, mentre una policy non equivariante richiede $O(|G| \, n_{tr})$ campioni per ottenere la stessa stima.*
>
> *Dimostrazione.* Il miglioramento di un fattore $|G|$ segue dal fatto che un modello equivariante usa implicitamente tutti i $|G|$ traslati di un dato campione come esempi di training (data augmentation esatta). Formalmente, si usa la decomposizione della loss in orbite di G: ogni orbita contribuisce una sola variabile di Rademacher, invece di $|G|$, nella complessità di Rademacher della classe di funzioni equivarianti. □

Per il cross-embodiment, l'ipotesi operativa è che la distribuzione delle traiettorie ottimali nel Common Action Space sia quasi-invariante rispetto alle trasformazioni di morfologia: compiti semanticamente equivalenti (prendere una bottiglia, aprire un cassetto) hanno strutture latenti simili. Questa invarianza latente è la giustificazione teorica dell'encoder morfologia-specifico $\varphi_i: A_i \to \mathbb{R}^{d_a}$ introdotto nel Capitolo 6, il quale apprende la mappa che porta la distribuzione specifica dell'embodiment i sulla distribuzione universale nel CAS.

6.4 Informazione di Fisher, curvatura del manifold delle policy e stabilità del fine-tuning

Il processo di fine-tuning di un modello cross-embodiment pre-addestrato su nuovi dati robotici è essenzialmente un problema di ottimizzazione su varietà di probabilità, in cui la metrica naturale non è quella euclidea ma quella di Fisher. La matrice di informazione di Fisher di una policy $\pi_\theta(a|o)$ rispetto ai parametri θ è:

$$F(\theta) = E_{o, a \sim \pi_\theta} [(\nabla_\theta \log \pi_\theta(a|o)) (\nabla_\theta \log \pi_\theta(a|o))^\top] \quad \in \mathbb{R}^{|\theta| \times |\theta|}$$

La distanza di Fisher tra due parametri θ e $\theta + \delta\theta$ è $ds^2 = \delta\theta^\top F(\theta)\, \delta\theta$, e corrisponde alla distanza di Kullback-Leibler al second'ordine: $KL(\pi_\theta \,\|\, \pi_{\theta+\delta\theta}) \approx \tfrac{1}{2}\, \delta\theta^\top F(\theta)\, \delta\theta$. Il metodo del Natural Policy Gradient (Kakade 2002) aggiorna i parametri nella direzione del gradiente naturale:

$$\theta_{t+1} = \theta_t - \alpha\, F^{-1}(\theta_t)\, \nabla_\theta L(\theta_t)$$

che corrisponde al gradiente steepest ascent nella metrica di Fisher. Il Natural Gradient ha la proprietà di invarianza rispetto a riparametrizzazioni: il passo di aggiornamento nel parametro $\theta' = \varphi(\theta)$ produce la stessa direzione di ottimizzazione del passo nel parametro θ originale, a differenza del gradiente euclideo standard.

Nel contesto del fine-tuning VLA, la matrice $F(\theta)$ è inaccessibile in forma esatta per modelli con miliardi di parametri. L'algoritmo Kronecker-Factored Approximate Curvature (K-FAC) approssima F con una struttura a blocchi Kronecker: per un layer lineare W con input a $\in$

$\mathbb{R}$^n e output s = Wa $\in$ $\mathbb{R}$^m, il blocco corrispondente di F è approssimato da:

$$F\_W \approx \hat{A} \otimes \hat{S}, \quad con \quad \hat{A} = E[a\, a^{\top}] \in \mathbb{R}^{n \times n}, \quad \hat{S} = E[\nabla\_s L\, (\nabla\_s L)^{\top}] \in \mathbb{R}^{m \times m}$$

dove $\otimes$ denota il prodotto di Kronecker. L'inversione del blocco si riduce all'inversione di due matrici di dimensione n×n e m×m, invece dell'intera matrice mn×mn: un risparmio computazionale di ordine $O(n^2m^2)$ vs $O(n^2+m^2)$. Sebbene K-FAC non sia direttamente implementabile per transformer di grandi dimensioni, le sue idee ispirano i metodi di second'ordine a basso rango (Shampoo, AdaHessian) che trovano applicazione nel fine-tuning dei modelli VLA.

6.4 Informazione di Fisher, curvatura del manifold delle policy e stabilità del fine-tuning

Il processo di fine-tuning di un modello cross-embodiment pre-addestrato su nuovi dati robotici è essenzialmente un problema di ottimizzazione su varietà di probabilità, in cui la metrica naturale non è quella euclidea ma quella di Fisher. Questa sezione completa la costruzione della geometria informazionale delle policy, con applicazioni dirette alla stabilità del fine-tuning e alla progettazione di ottimizzatori robusti.

Matrice di Informazione di Fisher e Metrica Riemanniana

La **matrice di informazione di Fisher** di una policy $\pi_\theta(a|o)$ rispetto ai parametri $\theta \in R^p$ è definita come:

$$F(\theta) = E_{(o,a)\sim\rho_\pi}\left[\nabla_\theta log\pi_\theta(a|o) \cdot \nabla_\theta log\pi_\theta(a|o)^{\top}\right] \in R^{p\times p}$$

dove $\rho_\pi(o,a) = d^\pi(o)\pi_\theta(a|o)$ è la distribuzione congiunta osservazione-azione indotta dalla policy π_θ e dalla distribuzione marginale degli stati d^π. La matrice $F(\theta)$ è semidefinita positiva per costruzione (come somma di prodotti esterni) ed è definita positiva se e solo se le log-likelihood $log\pi_\theta(a|o)$ hanno gradienti linearmente indipendenti in θ con probabilità 1.

La matrice di Fisher definisce una metrica riemanniana sullo spazio dei parametri Θ: la lunghezza infinitesimale di un cammino $\theta(t)$ in Θ rispetto alla metrica di Fisher è $\| \dot{\theta}\|_F = \sqrt{\dot{\theta}^\top F(\theta)\dot{\theta}}$. Questa metrica non è arbitraria ma ha un fondamento intrinseco nella geometria dell'informazione: misura la distanza di Kullback-Leibler tra distribuzioni vicine. Precisamente, per $\theta' = \theta + \delta\theta$ con $\|\delta\theta\|$ piccolo:

$$D_{KL}\left(\pi_\theta \parallel \pi_{\theta'}\right) = \frac{1}{2}\delta\theta^\top F(\theta)\delta\theta + O\left(\|\delta\theta\|^3\right)$$

Questa relazione è fondamentale: il fine-tuning che preserva la struttura della policy pre-addestrata è esattamente quello che minimizza la KL-divergenza rispetto ai pesi originali, ovvero quello che produce spostamenti piccoli nella metrica di Fisher.

Il Teorema di Cramér-Rao per Policy Stochastiche

Il **teorema di Cramér-Rao** nella sua forma per policy robotiche fornisce un lower bound sulla varianza di qualsiasi stimatore non distorto della funzione di valore $Q^\pi(o, a)$.

Teorema 6.2 (Cramér-Rao per la funzione di valore). *Sia* $\hat{Q}\left(o, a; D_n\right)$ *uno stimatore non distorto della funzione action-value* $Q^\pi(o, a) = E\left[\sum_t \gamma^t r_t | o_0 = o, a_0 = a, \pi\right]$ *basato su* n *traiettorie di training. Per ogni* $(o, a) \in O \times A$*:*

$$Var\left(\hat{Q}\left(o, a; D_n\right)\right) \geq \frac{1}{n}\left[F(\theta)^{-1}\right]_{(o,a),(o,a)}$$

dove il termine di destra è l'elemento diagonale corrispondente alla parametrizzazione (o, a) *della matrice di informazione di Fisher inversa.*

Dimostrazione. Il bound segue dall'applicazione del teorema di Cramér-Rao classico all'estimazione del parametro scalare $Q^\pi(o, a)$ come funzionale della policy π_θ. La regola di Fisher per la derivata del log-likelihood rispetto a parametri funzionali, combinata con la disuguaglianza di Cauchy-Schwarz sullo spazio L^2 delle funzioni di score,

dà il bound in termini di $F(\theta)^{-1}$. La costante $1/n$ riflette l'accelerazione radice-n della stima con campioni i.i.d. ▫

Gradiente Naturale e la sua Interpretazione Geometrica

Il **gradiente naturale** (Amari 1998) è la direzione di massima pendenza della loss nella metrica di Fisher, ovvero il gradiente corretto per la curvatura del manifold delle policy:

$$\widetilde{\nabla}_{\theta} L(\theta) = F(\theta)^{-1} \nabla_{\theta} L(\theta)$$

L'aggiornamento del gradiente naturale è $\theta_{t+1} = \theta_t - \alpha \widetilde{\nabla}_{\theta} L\left(\theta_t\right)$. L'interpretazione geometrica è la seguente: il gradiente euclideo $\nabla_{\theta} L$ misura la pendenza nella metrica euclidea su Θ, che dipende dalla parametrizzazione scelta e non riflette la struttura intrinseca dello spazio delle distribuzioni. Il gradiente naturale $\widetilde{\nabla}_{\theta} L$ è invece la direzione di massima pendenza nella metrica di Fisher, che è indipendente dalla riparametrizzazione: se $\phi = g(\theta)$ è una riparametrizzazione liscia, il gradiente naturale in ϕ produce lo stesso aggiornamento nella distribuzione di quello in θ.

Questa invarianza per riparametrizzazione è cruciale per il fine-tuning dei modelli VLA: la loss di task robotico può essere ottimizzata senza che il percorso di ottimizzazione dipenda dalla parametrizzazione specifica dei layer del transformer (normalizzazione, scaling, residual connections), che altrimenti può distorcere la geometria euclidea dello spazio dei pesi.

Proposizione 6.2 (Equivalenza TRPO-gradiente naturale). *L'aggiornamento di Trust Region Policy Optimization (TRPO, Schulman et al. 2015) per la policy* π_{θ} *con vincolo* $D_{KL}\left(\pi_{\theta_{old}} \| \pi_{\theta}\right) \leq \delta$ *è equivalente, nell'approssimazione al secondo ordine, a un passo di gradiente naturale con step-size* $\alpha = \sqrt{2\delta / \dot{\theta}^{\top} F(\theta) \dot{\theta}}$.

Dimostrazione. Il problema di TRPO è $max_{\theta} L(\theta)$ soggetto a $D_{KL}\left(\pi_{\theta_{old}} \| \pi_{\theta}\right) \leq \delta$. Approssimando L al primo ordine e D_{KL} al secondo ordine con $F\left(\theta_{old}\right)$, si ottiene il problema quadratico vincolato:

$max_{\delta\theta} \nabla L^{\top} \delta\theta$ soggetto a $\frac{1}{2}\delta\theta^{\top} F \delta\theta \leq \delta$. Le condizioni KKT danno $\delta\theta^{*} \propto F^{-1}\nabla L = \widetilde{\nabla}_{\theta} L$, che è il gradiente naturale. ▫

Curvatura del Manifold e Stabilità del Fine-Tuning

La curvatura del manifold delle policy, misurata tramite il tensore di curvatura di Riemann della metrica di Fisher, è direttamente legata alla stabilità del fine-tuning. Le regioni di alto rango di $F(\theta)$ corrispondono a parametri che controllano fortemente la distribuzione della policy, e pertanto sono le più sensibili a perturbazioni.

Per un transformer con architettura VLA, la struttura della matrice di Fisher può essere analizzata tramite la decomposizione per layer. Sia $F_l(\theta)$ la sotto-matrice di Fisher relativa al layer l. Si può mostrare che i layer più profondi (vicini all'output) hanno $\| F_l \|_2$ tipicamente maggiore dei layer superficiali, poiché i gradienti fluiscono più direttamente dall'output ai parametri profondi. Questo giustifica la pratica comune nel fine-tuning dei VLA di usare **learning rate decrescenti per profondità** (layer-wise learning rate decay): gli strati profondi richiedono step-size più piccoli per rimanere nella regione di stabilità della metrica di Fisher.

Corollario 6.2 (Bound sulla distorsione del fine-tuning con LoRA e metrica di Fisher). *Sia $\Delta W = BA$ un adattamento LoRA con rango r. L'aumento della KL-divergenza rispetto alla policy originale è bounded da:*

$$D_{KL}\left(\pi_{\theta+\Delta\theta} \,\|\, \pi_{\theta}\right) \leq \frac{1}{2}\Delta\theta^{\top} F(\theta)\Delta\theta + O\left(\|\Delta\theta\|^{3}\right) \leq \frac{1}{2}\|F(\theta)\|_2 \cdot \|\Delta\theta\|^{2}$$

Per LoRA con $\|\Delta W\|_F = O\left(r\sqrt{mn}\right)$ (Proposizione 5.2), il bound diventa $O\left(r^2 mn \|F_l\|_2\right)$, lineare nel rango r^2 e nella dimensione del layer.

Dimostrazione. Il primo bound segue dall'espansione di Taylor della KL-divergenza in termini della metrica di Fisher. Il secondo segue dalla disuguaglianza spettrale $\| v\|_F^2 \leq \|F\|_2 \|v\|^2$. La stima su $\|\Delta\theta\|^2$ segue dalla Proposizione 5.2. ▫

6.5 Helix e la Dualità Cervello-Midollo nella Robotica

Il sistema Helix (Figure/OpenAI, 2025) propone una realizzazione hardware-software della dualità Planner-Executor introdotta da Gemini Robotics, con una differenza architetturale rilevante: il Planner (denominato S2 per "System 2", in analogia con il framework di Kahneman) opera a livello di pensiero linguistico-simbolico su un orizzonte di secondi-minuti, mentre l'Executor (S1, "System 1") opera come un VLA compatto a 2.5B parametri che esegue le primitive motorie a 20 Hz senza coinvolgimento del Planner per ogni azione singola.

La separazione S1/S2 è motivata dalla latenza: il Planner S2, basato su un LLM da 70B+ parametri, ha una latenza di 200–500 ms per risposta, accettabile per decisioni strategiche ma incompatibile con il controllo reattivo. L'Executor S1 ha latenza di 50 ms ed è responsabile dell'esecuzione affidabile di primitive come "grasp", "place", "rotate", mentre S2 decide la sequenza e i parametri di queste primitive in risposta all'evoluzione della scena.

Il problema del grounding continuo

Un contributo tecnico rilevante di Helix è la gestione del grounding continuo: quando le condizioni della scena cambiano durante l'esecuzione di una primitiva S1 (un oggetto scivola, un altro robot interferisce), il sistema deve rilevare l'anomalia e aggiornare il piano S2 prima che l'azione fallisca. Questo richiede un monitor di esecuzione che confronta l'osservazione corrente con l'osservazione attesa dalla primitiva in corso:

$$e\_exec(t) = \| o\_obs(t) - o\_pred(t) \|\_F > \theta\_anomaly \rightarrow \text{trigger S2}$$

dove o_pred(t) è predetta dal World Model (§8.1–8.3). Quando l'anomalia viene rilevata, S2 riceve il contesto aggiornato e genera un nuovo piano in tempo reale, mentre S1 esegue una primitiva di "safe hold" (tenere la posizione corrente) in attesa del nuovo piano. Il sistema di monitoraggio è implementato come una rete neurale leggera (10M parametri) che stima la likelihood del frame corrente sotto il modello delle primitive eseguite.

Capitolo 7 — RoboVLMs: Tokenizzazione Multimodale e Generalizzazione

L'architettura RoboVLMs, sviluppata nel contesto di un consorzio di laboratori nel 2024, rappresenta il tentativo più sistematico di costruire un'unica architettura neurale in grado di controllare robot con morfologie radicalmente diverse attraverso una singola rete addestrata con l'intero corpus di dati disponibili. Il flusso di dati in un modello VLA parte dalla tokenizzazione delle osservazioni. Per un'osservazione visiva, l'encoder ViT suddivide l'immagine in patch di dimensione P×P pixel e produce una sequenza di token visivi v = (v₁, …, v_{(H/P)²}) ∈ ℝ^{d_vis}, poi proiettata nello spazio nascosto tramite una matrice W_{vis} ∈ ℝ^{d×d_vis}. La fusione multimodale avviene tramite concatenazione:

$$s = [v_1, \ldots, v\_\{N\_v\}, \langle SEP \rangle, l_1, \ldots, l\_L, \langle EOS \rangle] \in \mathbb{R}^{(N\_v + L + 2) \times d} \quad (7.1)$$

Il transformer processa questa sequenza con meccanismi di auto-attenzione causale. La profondità del trunk — numero di layer L, tipicamente 32–48 per modelli da 7–13B — determina la capacità del modello di costruire rappresentazioni semanticamente ricche. Il punto più delicato dell'architettura è il meccanismo che connette lo spazio degli embedding ℝ^d allo spazio d'azione continuo A ⊆ ℝ^{d_a}. Nella versione a token continui (π_0), la proiezione è esplicita tramite una rete di flow matching. Nella versione lineare più semplice, sia h_{out} ∈ ℝ^d l'hidden state dell'ultimo token d'azione prodotto dal transformer: la proiezione â = W_a h_{out} con W_a ∈ ℝ^{d_a×d} produce l'azione media, e la covarianza viene parametrizzata separatamente da una rete MLP.

> **Proposizione 7.1 (Approssimazione di basso rango e perdita di capacità).**
>
> *Sia $W \in \mathbb{R}^{m\times n}$ una matrice di pesi del transformer pre-addestrato. L'aggiornamento LoRA $\Delta W = BA$ con $B \in \mathbb{R}^{m\times r}$, $A \in \mathbb{R}^{r\times n}$, $r \ll \text{rank}(W)$, limita la perturbazione $\|\Delta W\|_2 \le \|B\|_2\|A\|_2$. La distanza dai pesi iniziali in norma di Frobenius è $\|\Delta W\|_F \le r \cdot \|B\|_2\,\|A\|_F$, dove il membro destro cresce come $O(\sqrt{r})$ per pesi casuali, indicando che la capacità di adattamento scala sublinearmente con il rango r.*

7.1 (espansione) Vision Transformer: Patch Embedding, Complessità e Teorema di Approssimazione

Patch Embedding e Tokenizzazione dell'Immagine

Il Vision Transformer (ViT, Dosovitskiy et al. 2020) tokenizza un'immagine $I \in R^{H \times W \times C}$ in una sequenza di $N = HW/P^2$ patch di dimensione $P \times P$ pixel. Ogni patch $p_k \in R^{P^2 C}$ viene proiettato in uno spazio embedding di dimensione d tramite una proiezione lineare apprendibile $E \in R^{(P^2 C) \times d}$:

$$z_k^{(0)} = p_k E + e_k^{pos}, \quad k = 1, \ldots, N$$

dove $e_k^{pos} \in R^d$ è il vettore di posizione del k-esimo patch. La scelta della dimensione del patch P determina un trade-off fondamentale: patch più piccoli ($P = 8$ o $P = 14$) producono sequenze più lunghe e rappresentazioni più dettagliate, a costo di una complessità quadratica nell'attenzione; patch più grandi ($P = 32$) riducono N ma perdono risoluzione spaziale.

Per i modelli VLA, la sequenza di N token visivi viene concatenata alla sequenza di T token linguistici per formare una sequenza totale di $N + T$ token su cui opera il transformer. La complessità dell'attenzione è $O\left((N + T)^2 d\right)$, che per $N = 256$ (immagine 224×224 con $P = 14$) e $T = 512$ è $O\left(768^2 d\right) \approx 590000d$ per layer. Con $d = 4096$ (LLaMA-3 7B), ogni layer di attenzione richiede circa 2.4×10^9 FLOP.

Teorema di Approssimazione per ViT

Proposizione 7.1 (Approssimazione universale dei ViT). *Sia $f: R^{H \times W \times C} \to R^d$ una funzione continua sull'immagine. Un ViT con L layer di attenzione, dimensione d, e h teste può approssimare f con errore uniforme ε usando $L = O\left(\varepsilon^{-2/(s+\alpha)}\right)$ layer, dove s è la dimensione intrinseca dell'immagine e α è il grado di regolarità di f (nell'ordine di Sobolev H^α).*

Dimostrazione (sketch). La prova segue dalla combinazione di due risultati: (1) il teorema di approssimazione per transformer di Yun et al. (2020) che mostra la densità delle funzioni realizzate da transformer nella classe delle funzioni continue su sequenze compatte; (2) il teorema di Kolmogorov-Arnold per la decomposizione di funzioni continue multivariabili in composizioni di funzioni univariate. La dipendenza dal grado di Sobolev α segue dall'analisi dell'errore di approssimazione lineare per funzioni in H^{α} tramite basi ortonormali (Fourier o wavelets), che è il meccanismo di codifica implicito nelle teste di attenzione. ▫

Confronto CLIP, SigLIP e DINOv2 come Backbone Visivi

I backbone visivi per i modelli VLA differiscono non soltanto nell'architettura ma nell'obiettivo di pre-addestramento, che determina la struttura geometrica dello spazio embedding.

CLIP (Radford et al. 2021) è addestrato con una loss di contrastive learning image-text su coppie $\left(I_i, t_i\right)$ di immagini e testi. L'obiettivo è la loss InfoNCE su un batch di N coppie:

$$L_{CLIP} = -\frac{1}{N}\sum_{i=1}^{N} log \frac{exp\left(\langle v_i, l_i\rangle/\tau\right)}{\sum_{j=1}^{N} exp\left(\langle v_i, l_j\rangle/\tau\right)}$$

dove $v_i = f_V\left(I_i\right)$ e $l_i = f_L\left(t_i\right)$ sono gli embedding visivi e linguistici normalizzati, e τ è la temperatura. La metrica indotta sullo spazio visivo di CLIP è quella del **coseno** (prodotto interno su S^{d-1}): feature di oggetti semanticamente simili sono vicine nella distanza coseno. Per i modelli VLA, questo significa che CLIP è ottimale per il riconoscimento semantico (che oggetto è?) ma meno preciso per la geometria (dove è? come è orientato?).

La similarità coseno CLIP R_CLIP(s◎, g) = cos(f_V(I(s◎)), f_L(g)) / τ ha proprietà metriche precise che influenzano il training RL. La metrica coseno su S^{d-1} (sfera unitaria d-dimensionale) non è una distanza euclidea, ma soddisfa la condizione di metrica di Riemannian manifold con curvatura sezionale costante κ = 1. Questo ha implicazioni per il gradiente del reward CLIP rispetto alle azioni: ∂R_CLIP/∂a = (1/τ) · (∂cos/∂f_V) ·

$(\partial f_V/\partial I) \cdot (\partial I/\partial a)$, dove il termine $\partial I/\partial a$ richiede la differenziabilità del renderer di stato $I(s_{\circledcirc})$ — raggiunta tramite differentiable rendering (§9.4) o approssimata con finite differences.

Il principale vantaggio del reward CLIP rispetto al reward scalare manuale è la generalizzazione zero-shot: il modello CLIP pre-addestrato su 400 milioni di coppie immagine-testo ha appreso una rappresentazione dello spazio visivo che allinea semanticamente immagini e descrizioni. Per una descrizione del goal g = 'robot in posizione eretta con braccia distese', la similarità CLIP con stati fisicamente distinti ma semanticamente simili (ad esempio, robot con diverse proporzioni in postura eretta) sarà alta, producendo un reward robusto alle variazioni irrilevanti della scena. Questo si contrappone ai reward basati su feature geometriche esplicitamente programmate, che richiedono un'ingegneria accurata delle invarianze desiderate.

Analisi della struttura del reward CLIP nel contesto del RL

Sia $p_{CLIP}(a \mid o, g) \propto \exp(R_{CLIP}(f(o,a), g) / \tau)$ la distribuzione di policy indotta dal reward CLIP. La temperatura τ controlla il grado di concentrazione: per $\tau \to 0$, la policy converge alla policy deterministica $\text{argmax}_a R_{CLIP}$; per $\tau \to \infty$, la policy converge alla distribuzione uniforme (massima entropia). Il training RL con reward CLIP equivale quindi a ottimizzare il variational lower bound su un obiettivo di maximum a posteriori: $\pi^* = \text{argmax}_\pi E_\pi[R_{CLIP}] - \tau \, KL(\pi \parallel p_{ref})$, dove p_{ref} è una policy di riferimento. Questa struttura è esattamente quella dell'algoritmo RLHF (§5.4): il reward CLIP gioca il ruolo del reward model umano, e la KL-penalizzazione previene il reward hacking.

> Proposizione 25.2 (Consistenza del reward CLIP come proxy del reward umano). Sia $R_{human}(s, g)$ la valutazione umana della qualità dello stato s rispetto al goal g. Se la distribuzione congiunta (R_{human}, R_{CLIP}) su $S \times G$ è tale che $R_{human} = f(R_{CLIP}) + \varepsilon$ con ε indipendente da R_{CLIP} e $E[\varepsilon] = 0$, allora la policy che massimizza $E[R_{CLIP}]$ massimizza anche $E[R_{human}]$ in aspettativa. La condizione ε indipendente da R_{CLIP} è soddisfatta se e solo se R_{CLIP} è un sufficiente statistic di R_{human}.

Dimostrazione. La decomposizione R_human = f(R_CLIP) + ε implica E[R_human | R_CLIP] = f(R_CLIP). La policy che massimizza E[R_CLIP] massimizza anche E[f(R_CLIP)] = E[E[R_human | R_CLIP]] = E[R_human] per il tower property dell'aspettazione. La condizione che ε sia indipendente da R_CLIP è equivalente alla sufficienza di R_CLIP per la distribuzione di R_human condizionata sullo stato-goal, che è la condizione di no-information-loss nella compressione R_human → R_CLIP. □

La non-stazionarietà del reward CLIP durante il training RL è un problema tecnico rilevante: i pesi dell'encoder visivo f_V sono aggiornati durante il fine-tuning del backbone (se si adotta il training end-to-end), modificando la funzione di reward durante il processo di ottimizzazione della policy. Questa co-evoluzione del reward e della policy crea dinamiche non stazionarie che possono portare a instabilità del training o a degenerate equilibria (stati in cui la policy massimizza il reward del fine-tuned CLIP ma non più la preferenza umana originale). La soluzione standard è congelare i pesi del backbone CLIP durante il training RL e aggiornare solo la policy — il che corrisponde all'assunzione di stazionarietà del reward nel framework MDP standard.

Recenti sviluppi (VPT — Video PreTraining, Baker et al. 2022; GROOT, §12.6) mostrano che è possibile usare i VLM non solo come funzioni di reward scalari ma come prior sulla distribuzione di traiettorie desiderabili. In questa formulazione, il reward CLIP diventa il logaritmo di un prior implicito sulla policy: $\log \pi\_CLIP(a \mid o, g) \propto R\_CLIP(f(o,a), g)$, e l'obiettivo di RL diventa massimizzare $KL(\pi \parallel \pi\_CLIP)$, equivalente a trovare la policy che si discosta minimalmente dalla prior CLIP nel senso della KL-divergenza mantenendo alto il valore atteso del task. Questo framework unifica il reward CLIP con l'IRL bayesiano e produce policy che sono simultaneamente performanti e semanticamente coerenti con il prior linguistico del goal.

Calibrazione della temperatura e tradeoff semantica-precisione

La scelta della temperatura τ nel reward $R\_CLIP = \cos(\cdot,\cdot)/\tau$ rappresenta un fondamentale tradeoff tra semantica e precisione geometrica. Per τ piccolo (reward sharply peaked), piccole differenze nella similarità CLIP

producono grandi differenze nel reward, spingendo la policy verso il manifold degli stati con massima similarità — che corrisponde alla interpretazione più letterale dell'immagine goal. Per τ grande (reward smooth), il reward varia lentamente con lo stato, producendo una funzione di reward densa che facilita il training RL ma riduce la discriminazione semantica. La calibrazione ottimale di τ si determina tramite la condizione che il segnale di gradient ∇_a R_CLIP abbia varianza sufficientemente alta da guidare l'esplorazione ma non tanto da saturare gli attuatori. Empiricamente, $\tau \in [0.07, 0.2]$ produce i migliori risultati su task manipolatori standard, corrispondente alle temperature usate nel training CLIP originale.

SigLIP (Zhai et al. 2023) sostituisce la loss softmax di CLIP con una loss **sigmoid** che tratta ogni coppia indipendentemente:

$$L_{SigLIP} = -\frac{1}{N^2}\sum_{i,j}\left[y_{ij}\,log\sigma\left(z_{ij}\right) + \left(1 - y_{ij}\right)log\left(1 - \sigma\left(z_{ij}\right)\right)\right]$$

dove $y_{ij} = 1$ se (i, j) è una coppia positiva, $z_{ij} = \langle v_i, l_j \rangle/\tau - b$ con b un bias apprendibile. La loss sigmoid elimina la dipendenza dalla dimensione del batch (che influenza il denominatore della softmax di CLIP) e produce embedding più uniformemente distribuiti sulla sfera S^{d-1}.

DINOv2 (Oquab et al. 2023) è addestrato con un obiettivo self-supervised che combina DINO (distillazione da insegnante a studente) con iBOT (masked image modeling). L'obiettivo non usa testo e produce feature che codificano la **struttura geometrica locale** dell'immagine — bordi, texture, profondità relativa — con una risoluzione spaziale superiore a CLIP e SigLIP. La metrica indotta è più vicina alla distanza Euclidea su patch locali che al coseno globale.

Proposizione 7.2 (Tradeoff geometria-semantica per backbone VLA). *Sia* $E_{sem}(\phi)$ *l'errore medio di riconoscimento semantico e* $E_{geo}(\phi)$ *l'errore di stima geometrica (posizione e orientazione) per un backbone* ϕ. *Empiricamente, esiste un tradeoff di Pareto tra i due errori parametrizzato dall'obiettivo di pre-addestramento: CLIP minimizza* E_{sem}, *DINOv2 minimizza* E_{geo}, *SigLIP è sul fronte di Pareto tra i due. La combinazione convessa*

$\phi_{mix} = (1 - \alpha)\phi_{CLIP} + \alpha\phi_{DINOv2}$ *nel feature space produce un errore congiunto* $E_{sem}(\phi_{mix}) + \lambda E_{geo}(\phi_{mix})$ *ottimale per* $\alpha^{*} = \lambda/(1 + \lambda)$ *nell'approssimazione lineare.*

Geometria della rappresentazione visiva e teoria dell'informazione

La struttura geometrica dello spazio embedding di un ViT pre-addestrato può essere analizzata tramite la dimensione intrinseca della varietà delle rappresentazioni. Sia Φ: ℝ^{H×W×C} → ℝ^d il backbone visivo e M_Φ = {Φ(I) : I ∈ data distribution} ⊆ ℝ^d la varietà delle rappresentazioni. La dimensione intrinseca dim_int(M_Φ) — stimabile tramite il correlation dimension o l'estimatore di Levina-Bickel — misura quanti gradi di libertà effettivi sono presenti nella distribuzione delle rappresentazioni. Per ViT addestrati su ImageNet, misurazioni empiriche (Anagnostopoulos et al. 2022) mostrano dim_int ≈ 20–40 per d = 768, indicando una compressione di oltre un ordine di grandezza rispetto alla dimensione nominale dello spazio embedding.

La dimensione intrinseca ha implicazioni dirette per il campionamento efficiente nel fine-tuning dei modelli VLA (§5.4): se la varietà delle rappresentazioni ha dim_int = k ≪ d, allora l'adattamento della policy richiede di modificare soltanto le componenti lungo le k direzioni principali della varietà — il che motiva formalmente l'approccio LoRA con rango r ≥ dim_int. La relazione tra il rango ottimale r* e la dimensione intrinseca k è data dalla proposizione seguente.

> Proposizione 7.3 (Rango ottimale di LoRA e dimensione intrinseca). Sia W ∈ ℝ^{m×n} una matrice di peso del transformer pre-addestrato e sia d_int la dimensione intrinseca della distribuzione degli input X al layer corrispondente. Il rango ottimale r* che minimizza l'errore di fine-tuning per un dato budget di parametri soddisfa r* ≥ d_int con alta probabilità. Specificamente, per ogni r < d_int esiste una distribuzione di task robotici per cui un adattamento LoRA di rango r ha errore asintoticamente maggiore dell'adattamento full-rank.

Dimostrazione. La prova usa il fatto che la distribuzione degli input al layer di peso W si concentra su una varietà di dimensione d_int. Per ogni aggiornamento $\Delta W = BA$ di rango $r < d_{int}$, esiste almeno una direzione nella varietà delle input che non è coperta dall'immagine di A (per il rank-nullity theorem: $\dim \ker(A^T) = n - r > n - d_{int} \geq 0$). Le variazioni della funzione di output lungo questa direzione non possono essere catturate da ΔW, producendo errore residuale non nullo. □

La connessione tra la geometria dello spazio embedding e la struttura del flusso di attenzione emerge tramite il concetto di attention entropy. L'entropia dell'attenzione $H_{att}(l) = -\Sigma_{i,j} A_{ij}^{(l)} \log A_{ij}^{(l)}$ al layer l misura quanto disperso è il pattern di attenzione: bassa entropia corrisponde ad attenzione concentrata su pochi token (attenzione sharp), alta entropia a attenzione distribuita uniformemente (attenzione diffusa). Per ViT addestrati con obiettivi di contrastive learning (CLIP), i layer profondi tendono ad avere attenzione sharp sui token semanticamente rilevanti, mentre per ViT addestrati con self-supervised masking (DINOv2), i pattern di attenzione sono più diffusi e correlati alla struttura locale dell'immagine. Questa differenza strutturale spiega il tradeoff semantica-geometria della Proposizione 7.2: attenzione sharp favorisce la semantica globale, attenzione diffusa favorisce la struttura spaziale locale.

Multi-scale ViT e gerarchie di patch per VLA ad alta risoluzione

L'architettura ViT standard usa un singolo livello di patch di dimensione fissa P×P. Per immagini robotiche ad alta risoluzione (480×640 pixel), P=14 produce $N = 480 \times 640 / 14^2 \approx 1560$ token, rendendo la complessità $O(N^2 d)$ dell'attenzione proibitiva per il deployment su hardware embedded. Le architetture multi-scale (Swin Transformer, Liu et al. 2021; MViT, Fan et al. 2021) superano questa limitazione tramite una gerarchia di window attention: al livello l, l'attenzione è calcolata all'interno di finestre locali di dimensione $w_l \times w_l$ che si espandono con la profondità. La complessità risultante è $O(N w^2 d)$ invece di $O(N^2 d)$, scalando linearmente con il numero di token. Per i modelli VLA con immagini robotiche, questo tradeoff è critico: la struttura locale-globale delle scene manipolative (dove le informazioni rilevanti sono sia la posa globale del robot sia i dettagli

locali dell'oggetto da manipolare) beneficia dell'architettura multi-scale più di quanto faccia la classificazione di scene naturali.

La teoria dell'approssimazione per ViT multi-scale estende la Proposizione 7.1 tenendo conto della struttura gerarchica. Per un ViT con L livelli di risoluzione $l_1 > l_2 > ... > l\_L$ (numero di token decrescente) e window size w_l al livello l, il numero di layer necessari per approssimare una funzione con struttura multiscala di grado di Sobolev α e scala caratteristica s_c è $L = O((s\_c/\varepsilon)^{1/\alpha})$ invece di $O((w/\varepsilon)^{1/\alpha})$ per il ViT standard, dove w è la dimensione dell'intera immagine. Il risparmio di $O((w/s\_c)^{1/\alpha})$ layer è significativo per immagini robotiche in cui s_c ≈ diametro tipico degli oggetti manipolati (10–100 pixel su 640 totali), producendo una riduzione di 4–40× nel numero di layer necessari.

7.2 Attenzione cross-modale e fusione visio-linguistica

La fusione dei token visivi e linguistici nel trunk del transformer avviene tramite il meccanismo di attenzione full-attention (tutti i token si prestano attenzione a tutti gli altri) nella versione standard, o tramite cross-attention dedicata nella variante con encoder-decoder separato. Quest'ultima architettura — adottata ad esempio in Flamingo e nelle versioni più recenti di RoboVLMs — elabora i token visivi con un visual encoder congelato (ViT o SigLIP) e inietta le rappresentazioni visive nelle layer del decoder linguistico tramite layer di cross-attention inseriti ogni k layer.

Formalmente, un layer di cross-attention riceve come query i token linguistici $x\_L \in \mathbb{R}^{L\times d}$ e come chiavi e valori i token visivi proiettati $x\_V \in \mathbb{R}^{N\_V\times d}$:

$$x\_L' = x\_L + CrossAttn(x\_L\, W^Q,\ x\_V\, W^K,\ x\_V\, W^V)$$

La proiezione dei token visivi nello spazio delle chiavi tramite $W^K \in \mathbb{R}^{d\_vis\times d\_k}$ consente di adattare la dimensionalità dell'encoder visivo (tipicamente $d\_vis = 768$ o 1024 per ViT-B e ViT-L) alla dimensionalità nascosta del decoder linguistico ($d = 4096$ per 7B). Il vantaggio rispetto alla semplice concatenazione (eq. 7.1 della monografia) è la possibilità di congelare i pesi del decoder linguistico e addestrare solo le layer di

cross-attention, riducendo il numero di parametri addestrabili e minimizzando il catastrophic forgetting.

Un problema specifico della cross-attention nel contesto robotico è il token bottleneck: se N_V è grande (ad esempio N_V = (224/14)² = 256 patch per un'immagine 224×224 con patch 14×14 in ViT-B/14), il costo della cross-attention è O(L N_V d_k) per passo, che può diventare proibitivo per sequenze linguistiche lunghe L $\gg$ 1. La compressione dei token visivi tramite mean pooling o tramite un layer di Perceiver Resampler (come in Flamingo) riduce N_V a un numero fisso K ≤ 64, a costo di una perdita di risoluzione spaziale. La scelta ottimale di K dipende dal trade-off tra ricchezza della rappresentazione visiva e costo computazionale, e viene tipicamente determinata per ricerca su griglia su un insieme di task di validazione.

7.2 (espansione) VQ-VAE per la Tokenizzazione delle Azioni: Teoria della Quantizzazione Vettoriale

VQ-VAE e Codebook Learning

Il **VQ-VAE** (van den Oord et al. 2017) è un autoencoder variazionale con quantizzazione vettoriale discreta: il vettore latente $z_e = E_\phi(x)$ viene mappato nel codebook $\{e_1, \ldots, e_K\} \subset R^d$ tramite la regola del nearest-neighbor:

$$z_q = e_k, \quad k = arg \min_{j \in \{1,\ldots,K\}} \| z_e - e_j \|$$

La quantizzazione è non differenziabile (argmin ha gradiente zero), ma il gradiente è approssimato tramite straight-through: il gradiente di z_q rispetto ai parametri dell'encoder è calcolato come se $z_q = z_e$. La loss di addestramento è:

$$L_{VQ} = \|x - D_\psi(z_q)\|^2 + \|sg(z_e) - z_q\|^2 + \beta\|z_e - sg(z_q)\|^2$$

$\hookrightarrow reconstruction$ $\hookrightarrow codebook$ $\hookrightarrow commitment$

Il termine di codebook aggiorna i vettori del codebook verso gli encoder outputs, il termine di commitment impedisce che z_e si allontani dai vettori del codebook (con $\beta \approx 0.25$).

Per la tokenizzazione delle azioni nei modelli VLA, il VQ-VAE viene applicato alle traiettorie di azioni per produrre una sequenza di token discreti. Una traiettoria di T azioni $\{a_1, ..., a_T\}$ diventa una sequenza di indici di codebook $\{k_1, ..., k_{T'}\}$ con $T' \leq T$ (possibile downsampling temporale). Il modello VLA autoregressivo genera poi i token di azione discreti uno per uno, come fa per i token linguistici.

Teorema della Distorsione-Tasso per il Codebook Ottimale

La qualità della tokenizzazione con VQ-VAE è misurata dalla **distorsione** $D = E\left[\| z_e - z_q \|^2\right]$ (errore di quantizzazione medio) in funzione del **tasso** $R = log_2 K$ (bit per token). La teoria della quantizzazione vettoriale fornisce il codebook ottimale.

Teorema 7.1 (Distorsione ottimale con K centri, Bennett 1948 generalizzato). *Sia $p(z)$ la distribuzione della codifica z_e con densità liscia su R^d. Il codebook ottimale di K vettori che minimizza la distorsione soddisfa:*

$$D^* \sim C(d) \cdot K^{-2/d} \cdot \left(\int_{R^d} p(z)^{d/(d+2)} dz \right)^{(d+2)/d}$$

per $K \to \infty$, dove $C(d) = d/(d+2) \cdot (2\pi e)^{-1}$ è la costante di Bennett. La distorsione decade come $K^{-2/d}$: per raddoppiare la precisione ($D^* \to D^*/4$), occorre moltiplicare K per 2^d.

Dimostrazione. La dimostrazione usa la condizione di ottimalità per codebook di Zador (1982): il codebook ottimale distribuisce i centri con densità di quantizzazione $\lambda(z) \propto p(z)^{d/(d+2)}$ (proporzionale alla densità di input elevata alla potenza $d/(d+2)$). La stima della distorsione per ogni cella di Voronoi con densità λ uniforme e volume $1/K$ è $d(d+2)^{-1}(K\lambda(z))^{-2/d}$. Integrando su tutto lo spazio si ottiene l'espressione del teorema. ▫

La conseguenza pratica per i modelli VLA è che con $K = 512$ token di codebook e dimensione latente $d = 256$, la distorsione $D^* = O\left(512^{-2/256}\right) = O(0.95)$ è ancora relativamente alta. Per raggiungere distorsioni competitive con la rappresentazione continua, il VQ-VAE deve usare residual quantization (VQ gerarchico a più livelli) o aumentare la dimensione del codebook a $K \geq 4096$.

Residual Quantization e codebook gerarchico

La residual quantization (RQ) supera il limite della distorsione del VQ-VAE semplice applicando la quantizzazione in cascata: il residuo dell'errore di quantizzazione al livello l viene quantizzato al livello l+1, fino a L livelli totali. Formalmente, sia z_e il vettore latente da quantizzare. Il primo livello produce z_q^(1) = nearest(z_e, C^(1)) con residuo r^(1) = z_e − z_q^(1). Il secondo livello quantizza il residuo: z_q^(2) = nearest(r^(1), C^(2)) con residuo r^(2) = r^(1) − z_q^(2). L'approssimazione totale è z_RQ = Σ_{l=1}^L z_q^(l), con codice (k_1,...,k_L) ∈ {1,...,K}^L. La distorsione della RQ con L livelli e K centri per livello soddisfa D_RQ^(L) ≤ (K^{−2/d})^L = K^{−2L/d}: la distorsione decade esponenzialmente nel numero di livelli L, permettendo di raggiungere distorsioni competitive con la rappresentazione continua con L = O(d log K / 2) livelli.

Il numero di bit per token in una RQ a L livelli con K centri è $\log_2(K^L) = L \log_2(K)$. Per L = 8 e K = 256, si ha $\log_2(256^8) = 64$ bit per token — paragonabile alla rappresentazione float64 ma con struttura discreta che consente il campionamento autoregressivo del modello VLA. Questa osservazione è alla base dei recenti modelli RVQ-VLA (Residual Vector Quantized VLA, Shi et al. 2024): le azioni continue sono quantizzate con RQ prima del tokenizzatore linguistico, producendo sequenze di token discreti che il transformer genera autoregressivamente. La qualità di ricostruzione con L = 8 livelli supera quella della quantizzazione scalare a 8 bit per dimensione (metodo tradizionale di tokenizzazione discreta delle azioni), mantenendo la coerenza semantica dello spazio latente.

Teoria dell'informazione del codebook: capacità e ridondanza

La capacità informativa di un codebook di K vettori in $\mathbb{R}^d$ sotto la distribuzione degli input p(z) è la quantità di informazione mutua I(Z;

Z_q) tra il latente originale Z e il latente quantizzato Z_q = nearest(Z, C). Per distribuzione gaussiana isotropica Z ~ N(0, σ^2 I_d) e codebook ottimale di K vettori, questa mutua informazione soddisfa I(Z; Z_q) = h(Z) − h(Z | Z_q) = d/2 log(2πe σ^2) − E[h(Z | Z_q = k) | k], dove h(Z | Z_q = k) è l'entropia differenziale del cluster k-esimo. Per cluster approssimativamente gaussiani con varianza σ_k^2 (derivante dalla distorsione locale D_k), si ha I(Z; Z_q) ≈ d/2 log(σ^2/σ_k^2) = d/2 log(SNR_k), dove SNR_k = σ^2/D_k^{2/d} è il rapporto segnale-rumore del cluster k.

La mutua informazione I(Z; Z_q) è direttamente connessa alla qualità delle azioni generate dal modello VLA: alta mutua informazione significa che Z_q preserva molte delle informazioni di Z, il che implica che le azioni generate dal decoder del VLA sono fedeli all'intenzione originale. Per il theorem della compressione di Shannon, la minima distorsione raggiungibile con un codebook di K vettori in d dimensioni è D* = σ^2 K^{−2/d}, che corrisponde alla massima mutua informazione I* = d/2 log K^{2/d} = log K. Questo bound è raggiunto dal codebook ottimale di Zador (Teorema 7.1) ed è il limite teorico per qualsiasi schema di tokenizzazione discreta delle azioni.

> Corollario 7.2 (Numero minimo di bit per token per qualità robotica). Per una distribuzione di azioni robotiche con dimensione intrinseca d_int (Proposizione 7.3) e precisione richiesta ε (errore di tracking dell'end-effector), il numero minimo di bit B* per token di azione soddisfa B* ≥ d_int/2 · $\log_2((\sigma\_action/\varepsilon)^2)$, dove σ_action è la deviazione standard delle azioni nella distribuzione dei dati.
>
> Dimostrazione. La distorsione richiesta è D ≤ ε^2, quindi K ≥ (σ_action/ε)^d_int per il bound di Zador. Il numero di bit è B = $\log_2$(K) ≥ d_int $\log_2(\sigma\_action/\varepsilon)$. Per un braccio robotico con σ_action ≈ 0.1 m e ε = 1 mm, si ottiene B ≥ d_int · $\log_2$(100) ≈ 6.6 d_int. Per d_int = 6 (spazio se(3) del twist dell'end-effector), B ≥ 40 bit per token — giustificando i codebook da K ≥ 2^40/6 ≈ 2^7 = 128 usati nella pratica. □

La ridondanza del codebook è la frazione di codeword non utilizzate nella distribuzione dei dati di training: R = 1 − H(Z_q)/$\log_2$(K), dove H(Z_q) è l'entropia di Shannon del quantizzato. Una ridondanza alta (R vicino a 1)

indica che molti codeword sono usati raramente o mai, un fenomeno noto come codebook collapse nelle architetture VQ-VAE con training instabile. Il codebook collapse si manifesta tipicamente nelle prime fasi del training quando i vettori del codebook non coprono uniformemente la distribuzione dei latenti: l'aggiornamento del codebook (termine di codebook loss) è dominato dai cluster vicini alla media, lasciando i cluster periferici inutilizzati. La soluzione standard (codebook reset, van den Oord 2017) reinizializza i codeword inutilizzati con campioni casuali dal mini-batch corrente, garantendo la copertura distribuita del codebook a regime.

7.3 Scaling laws per la robotica: dati, parametri e generalizzazione

Le leggi di scaling (scaling laws) quantificano empiricamente come la performance di un modello di apprendimento dipende dalla dimensione del modello N_θ (numero di parametri), dalla quantità di dati di training D e dalla potenza di calcolo $C = O(N_\theta D)$. Per i modelli linguistici, Hoffmann et al. (2022, Chinchilla) hanno mostrato che la perplexity segue la legge di potenza:

$$L(N_\theta, D) = E + A / N_\theta^\alpha + B / D^\beta$$

con esponenti $\alpha \approx \beta \approx 0.5$ e costante irriducibile E (entropy della distribuzione dei dati). Il risultato chiave è che il modello ottimale a parità di compute $C = 6 N_\theta D$ ha $N_\theta \propto \sqrt{C}$ e $D \propto \sqrt{C}$: il compute deve essere distribuito equamente tra modello e dati. Per un compute $C = 10^{23}$ FLOPs, il modello ottimale ha circa 70B parametri e 1.4T token.

Le scaling laws per la robotica seguono la stessa struttura, ma con metriche di performance diverse (successo sul task, errore di tracking) e con la complicazione che i dati robotici hanno struttura eterogenea (robot diversi, task diversi, qualità delle dimostrazioni variabile). Un primo tentativo sistematico di misurazione è quello di Padalkar et al. (2023) nell'ambito del dataset Open X-Embodiment: raccogliendo 22 dataset robotici con 527.000 traiettorie su 23 robot diversi, si è misurata la performance in funzione delle dimensioni del modello (RT-1 con 35M parametri vs RT-2 con 55B parametri). Il risultato è un incremento di

successo da 53% a 67% su task non visti, consistente con una legge di scaling del tipo successo $\propto N_\theta^\gamma$ con $\gamma \approx 0.1$ (logaritmicamente lenta), molto più piatta delle scaling laws linguistiche.

L'interpretazione di questa differenza è che i dati robotici — acquisiti in ambienti fisici con alta variabilità — hanno una struttura di distribuzione più complessa rispetto al testo, con una coda pesante di situazioni rare (edge cases) che richiedono quantità enormi di dati per essere coperte adeguatamente. Questo spiega l'enorme enfasi sulla raccolta di dati su larga scala e sulla data augmentation tramite simulatori nei sistemi robotici moderni.

La legge di scaling ottimale per i modelli VLA deve includere un terzo parametro: la quantità di dati robotici D_r rispetto ai dati di pre-training linguistico-visivo D_l. Empiricamente, il rapporto ottimale $D_r / D_l \approx 10^{-3} – 10^{-2}$: troppo pochi dati robotici non sono sufficienti per adattare il modello; troppi annullano la conoscenza pre-addestrata per catastrophic forgetting. Questo trade-off è la ragione per cui tecniche di fine-tuning preservativo come LoRA, elastic weight consolidation e replay buffer sono così critiche per i modelli VLA.

La complessità campionaria del fine-tuning VLA può essere analizzata attraverso la teoria di Vapnik-Chervonenkis (VC). La classe di funzioni implementate da un transformer con N_θ parametri e pesi in $[-B, B]$ ha dimensione VC $\Omega(N_\theta \log(N_\theta/B))$ per l'architettura completamente connessa, ma tale stima non cattura la struttura sparsa degli aggiornamenti LoRA. Per adattamenti LoRA di rango r, la dimensione VC effettiva è ridotta a $O(r(m+n) \log(|V|/B))$ per layer, dove $|V|$ è il numero di pesi effettivamente modificati. Questo risultato giustifica formalmente la pratica di limitare r a valori dell'ordine di 4-64, ottenendo generalizzazione su nuovi compiti robotici con centinaia (anziché migliaia) di dimostrazioni.

7.3 (espansione) Generalizzazione Zero-Shot e Transfer Learning nei RoboVLMs

Bound di Generalizzazione per Transfer Learning

Il trasferimento da un modello linguistico-visivo pre-addestrato a una policy robotica coinvolge due distribuzioni: la distribuzione di pre-addestramento D_{pre} (coppie immagine-testo da internet) e la distribuzione di fine-tuning D_{robot} (coppie osservazione-azione da teleoperazione). La teoria del transfer learning fornisce bound espliciti sulla differenza tra l'errore sul dominio robotico e quello sul dominio di pre-addestramento.

Teorema 7.2 (Ben-David et al. 2010, applicato a VLA). *Sia $h: O \to A$ l'ipotesi appresa dal fine-tuning. Denotando con $\varepsilon_{robot}(h)$ e $\varepsilon_{pre}(h)$ gli errori attesi sui rispettivi domini:*

$$\varepsilon_{robot}(h) \leq \varepsilon_{pre}(h) + d_{H\Delta H}\left(D_{pre}, D_{robot}\right) + \lambda^{*}$$

dove $d_{H\Delta H}$ è la distanza $H\Delta H$ (symmetric difference divergence) tra i domini, e $\lambda^{*} = min_{h \in H}\left[\varepsilon_{robot}(h) + \varepsilon_{pre}(h)\right]$ è l'errore del miglior classificatore congiunto.

Dimostrazione. La bound segue dalla decomposizione del rischio sul dominio target in tre termini: il rischio sul dominio sorgente, la discrepanza tra le distribuzioni (misurata dalla distanza $H\Delta H$ che è la variazione totale massimizzata sulla classe di ipotesi), e l'errore del miglior ipotesi condivisa. I dettagli della dimostrazione sono in Ben-David et al. (2010), Teorema 2. ▫

Per i modelli VLA, il termine $d_{H\Delta H}\left(D_{pre}, D_{robot}\right)$ è la quantità critica: misura quanto le feature del backbone visivo pre-addestrato su immagini internet si trasferiscono alle osservazioni robotiche. Empiricamente, si osserva che backbone addestrati su dataset più diversificati (LAION-5B per CLIP, JFT-3B per SigLIP) producono distanze $d_{H\Delta H}$ minori e quindi bound di trasferimento migliori.

PARTE III

World Models Fisicie Simulazione Differenziabile

Capitolo 7.4 — Architetture VLA di Terza Generazione e Generalizzazione Composizionale

Raccordo con la monografia: Cap. 5–7 (RT-2, OpenVLA, pi0, RoboVLMs, scaling laws).

3.1 RoboPoint: Ragionamento Spaziale 3D nei VLA

I modelli VLA della seconda generazione (RT-2, OpenVLA, pi0) producono azioni come sequenze di token che codificano pose dell'end-effector in coordinate di immagine 2D o in frame robot. RoboPoint (Yuan et al., 2024) introduce una rappresentazione intermedia esplicita: prima di generare l'azione, il modello predice un punto di interesse 3D (waypoint) nell'immagine, poi genera l'azione che porta l'end-effector a quel punto. Questa decomposizione -- denominata point-conditioned action generation -- migliora significativamente le prestazioni su task che richiedono precisione geometrica (es. inserimento in slot, apertura di cassetti con maniglie piccole).

Formalmente, sia I l'immagine osservativa e g la descrizione del task. RoboPoint genera in due fasi:

$$p^* = \text{argmax}_p \, P_{VLM}(p \mid I, g) \quad \text{(waypoint prediction)}$$

$$a^* = \text{argmax}_a \, P_{action}(a \mid I, p^*, g) \quad \text{(action generation conditioned on waypoint)}$$

Il waypoint p^* e una coppia (u, v) di coordinate di immagine, eventualmente estesa a 3D tramite unprojection con la depth map. La rete di waypoint prediction e addestrata con supervisione debole: i waypoint di training vengono estratti automaticamente dalle dimostrazioni localizzando il punto di contatto previsto dalla cinematica diretta. Risultati su benchmark LIBERO-Spatial mostrano miglioramenti del 15-25% su task geometricamente precisi rispetto a OpenVLA standard.

3.2 Octo 2 e l'Architettura con Testa di Azione Multi-Modale

Octo 2 (Team Octo, 2024) introduce una testa di azione separata dal trunk del transformer, specializzata nella generazione di azioni per morfologie diverse. Invece di tokenizzare le azioni come testo (approccio RT-2/OpenVLA), Octo 2 usa una testa di azione basata su diffusione con architettura transformer separata che riceve i token di contesto dal trunk principale e genera azioni continue tramite un processo di denoising rapido (5 passi). La separazione trunk/testa consente di addestrare la testa di azione su dati robotici abbondanti senza pre-training linguistico, preservando la conoscenza semantica del trunk.

La loss di training di Octo 2 e:

$$L_{Octo2}(\theta) = L_{VLM}(\theta_{trunk}) + \alpha \cdot L_{diffusion}(\theta_{head} \mid h_{trunk})$$

dove h_trunk sono i token di output del trunk congelato. Questo approccio riduce il catastrophic forgetting rispetto al full fine-tuning (Proposizione 5.2) di un fattore proporzionale al numero di parametri della testa rispetto al trunk (tipicamente 10-20x meno parametri).

3.3 Generalizzazione Composizionale: Teoria e Misura

La generalizzazione composizionale -- la capacita di combinare skills apprese separatamente per eseguire nuovi task -- e la proprieta piu importante e piu difficile da misurare dei VLA generalisti. Un sistema dotato di generalizzazione composizionale perfetta potrebbe, avendo appreso separatamente "prendi la mela" e "metti nel cestino", eseguire "prendi la mela e mettila nel cestino" senza training aggiuntivo.

Il framework formale per misurare la generalizzazione composizionale e il SCAN benchmark (Lake & Baroni, 2018), adattato alla robotica come RoboSCAN: si costruisce un training set T di coppie (descrizione, traiettoria) e un test set V che include combinazioni di concetti presenti in T ma non viste come coppia. La compositionality gap e:

$$CG = 1 - (\text{successo}_V / \text{successo}_T)$$

Un sistema con CG = 0 e perfettamente composizionale; CG = 1 indica zero generalizzazione composizionale. I VLA correnti mostrano CG tipicamente nell'intervallo [0.3, 0.6]: una riduzione di 2-3x rispetto ai modelli puramente linguistici su task analoghi, ma ancora lontani dalla composizionalita umana (CG ~ 0.05).

3.4 Meccanismi di Composizionalita: Binding e Routing Neurale

La composizionalita emerge in due meccanismi distinti nei transformer. Il binding collega le proprieta a gli oggetti tramite l'attenzione cross-modale: il meccanismo di attenzione in (eq. 5.1) apprende a costruire rappresentazioni che codificano il colore "rosso" come attributo dell'oggetto "mela" invece che come concetto indipendente. Il routing seleziona quale subset di neuroni processa ogni concetto, implementando un meccanismo di mixture of experts implicito.

La teoria degli slot di attenzione (Locatello et al., 2020) formalizza il binding come una decomposizione della rappresentazione dell'immagine in K slot, ciascuno corrispondente a un oggetto:

$$h_{scene} = \sum_{k=1}^{K} \alpha_k * slot_k$$

dove alpha_k sono pesi di attenzione. La composizionalita e garantita se i slot sono appresi in modo indipendente: slot_k codifica le proprieta dell'oggetto k indipendentemente dalla posizione degli altri oggetti. L'integrazione del meccanismo slot-attention nei VLA e un'area di ricerca attiva con i sistemi

SLATE-VLA (2025) che mostrano CG ridotto del 15-20% rispetto ai transformer standard.

Capitolo 8 — Dreamer V3 e l'Architettura RSSM

I world models sono modelli generativi dell'ambiente che un agente può sfruttare per pianificare senza interagire direttamente con il mondo reale, apprendendo la dinamica dell'ambiente da esperienze passate e usando tale conoscenza per simulare il futuro durante l'ottimizzazione della policy. L'architettura Dreamer, nella sua terza iterazione (Dreamer V3, Hafner et al., 2023), rappresenta il punto di arrivo di una linea di ricerca avviata nel 2019 che ha progressivamente consolidato la capacità dei world models di gestire ambienti complessi con orizzonti temporali lunghi e ricompense scarse.

L'intuizione fondamentale di Dreamer è la separazione della rappresentazione latente in due componenti: uno stato deterministico h_t ∈ ℝ^{d_h} (la memoria ricorrente del sistema, aggiornata da una GRU) e uno stato stocastico z_t ∈ Z (un vettore di variabili discrete o continue campionate da una distribuzione latente). Questa separazione, denominata Recurrent State Space Model (RSSM), consente di addestrare il world model tramite backpropagation attraverso il tempo, mentre la componente stocastica cattura l'aleatorietà del mondo.

8.1 Il Modello RSSM: Struttura Formale

Il RSSM è definito da quattro componenti funzionali. Il transition model (prior) predice lo stato latente futuro dato lo stato corrente e l'azione, senza l'osservazione corrente: p_θ(z_{t+1} | h_{t+1}) dove h_{t+1} = f_θ(h_t, z_t, a_t) è l'aggiornamento ricorrente deterministico. Il representation model (posterior) aggiorna la stima dello stato incorporando l'osservazione: q_φ(z_t | h_t, o_t). Il decoder ricostruisce l'osservazione: p_θ(o_t | h_t, z_t). Il reward model predice la ricompensa: p_θ(r_t | h_t, z_t). Le equazioni del modello sono:

$$h\_t \;=\; f\_\theta(h\_\{t-1\},\, z\_\{t-1\},\, a\_\{t-1\}) \quad [GRU\ deterministica] \quad (8.1)$$

$$z\_t \sim p\_\theta(z\_t \mid h\_t) \quad [Prior\ stocastico] \quad (8.2)$$

$$z\_t \sim q\_\varphi(z\_t \mid h\_t,\, o\_t) \quad [Posterior\ di\ rappresentazione] \quad (8.3)$$

In Dreamer V3, z_t è discretizzato come una variabile categorica: z_t ∈ {0,1}^{32×32}, ovvero 32 variabili categoriche ognuna su 32 classi. La scelta di variabili discrete è motivata da due ragioni: la robustezza alla multimodalità (stati fisicamente distinti sono rappresentati da codici discreti distinti, evitando il posterior collapse delle VAE gaussiane) e la maggiore espressività per modellare transizioni di stato nette.

8.2 L'Obiettivo ELBO e la Trasformazione Symlog

Il training del world model massimizza la Evidence Lower BOund (ELBO) della log-verosimiglianza marginale delle sequenze di osservazioni. Per una traiettoria τ = (o_{1:T}, a_{1:T}, r_{1:T}), l'ELBO si scrive:

$$L_{WM}(\theta, \varphi) = E_{q\_\varphi}\Big[\sum_{t=1}^{T} log\ p\_\theta(o\_t \mid h\_t,\ z\_t) + log\ p\_\theta(r\_t \mid h\_t,\ z\_t) - \beta\, KL(q\_\varphi(z\_t \mid$$

(8.4)

Il contributo più rilevante di Dreamer V3 rispetto alle versioni precedenti è l'introduzione del KL balancing con free bits: il termine KL viene calcolato come max(KL(q ∥ p), f) per un valore f > 0, in modo da evitare che il modello collassi lo stato latente a una distribuzione degenere all'inizio del training. L'altro contributo è la trasformazione symlog delle osservazioni e delle ricompense: symlog(x) = sign(x) · log(|x| + 1). Questa trasformazione gestisce distribuzioni con code pesanti senza richiedere la conoscenza a priori della scala dei valori, e la funzione di loss del decoder diventa:

$$L_{dec}(o_t,\ \hat{o}\_t) = \|symlog(o_t) - \hat{o}\_t\|^2 \quad (8.5)$$

8.3 Pianificazione nel Modello del Mondo tramite Actor-Critic

Una volta addestrato il world model, la policy viene ottimizzata tramite simulazione immaginata: l'agente simula traiettorie di durata H nel world model e ottimizza la policy rispetto alla ricompensa tramite un algoritmo actor-critic. L'actor π_ψ(a_t | h_t, z_t) e il critic V_ξ(h_t, z_t) sono reti neurali i cui parametri sono ottimizzati sulle traiettorie latenti generate. Il valore di uno stato latente è il ritorno atteso scontato:

$$V(h\_t,\ z\_t) = E\_\pi\Big[\sum_{k=0}^{H-1} \gamma^k r_{t+k}\Big] \quad (8.6)$$

Il critic è addestrato minimizzando l'errore di stima del valore tramite il target λ-return, mentre l'actor è aggiornato tramite policy gradients reparametrizzati (straight-through gradients per variabili discrete, reparametrization trick per gaussiane). L'intero training è differenziabile attraverso H passi di simulazione immaginata, consentendo al gradiente di fluire dal valore accumulato alle azioni dell'actor.

8.4 Gradiente straight-through per variabili categoriche e suo fondamento teorico

In Dreamer V3, lo stato latente z_t è una variabile categorica discreta: z_t ∈ {e_1, ..., e_K}^{32} è un insieme di 32 variabili categoriche su K = 32 classi ciascuna, per un totale di 32 × K = 1024 variabili binarie. La variabile categorica i-esima ha distribuzione parametrizzata dai logit ψ_i ∈ ℝ^K: z_i ~ Categorical(softmax(ψ_i)). Il campionamento da una distribuzione categorica è un'operazione non differenziabile: la mappa ψ ↦ argmax_k softmax(ψ)_k ha gradiente zero quasi ovunque.

L'estimatore straight-through (STE) di Bengio et al. (2013) sostituisce il gradiente della funzione non differenziabile con il gradiente di una sua approssimazione differenziabile durante il backward pass, mentre usa il campionamento esatto durante il forward pass. Per una variabile categorica, il forward pass campiona z_i = one_hot(argmax softmax(ψ_i)) ∈ {0,1}^K, mentre il backward pass calcola il gradiente come se z_i = softmax(ψ_i) (la versione continua):

$$\partial z\_i/\partial \psi \leftarrow \partial[softmax(\psi\_i)]/\partial \psi = diag(softmax(\psi\_i)) - softmax(\psi\_i) \; softmax(\psi\_i)^{\top}$$

La giustificazione teorica del gradiente straight-through è la sua interpretazione come estimatore del gradiente della funzione di valore atteso ∂E_{z~p_ψ}[f(z)]/∂ψ. Per una funzione f(z) lipschitziana con costante L_f, l'errore dell'estimatore STE rispetto al gradiente corretto è:

$$\| STE - \nabla\_\psi \, E[f(z)] \| \le L\_f \cdot E[\| z - softmax(\psi) \|]$$

che decresce quando la distribuzione softmax è quasi concentrata su un'unica categoria (temperatura bassa), e aumenta nella regione di alta

entropia. Il comportamento dell'STE durante il training di Dreamer V3 è quindi auto-regolante: all'inizio, con alta entropia, il bias è grande ma il gradiente è informativo; a convergenza, con bassa entropia, il bias è piccolo e l'estimatore è accurato.

Una variante più precisa del gradiente straight-through è la tecnica Gumbel-Softmax con temperatura τ. Il campionamento $z_i \sim \text{Categorical}(\pi_i)$ è approssimato dalla variabile continua:

$$z_i^{(\tau)} = \text{softmax}((\log \pi_i + g_i) / \tau) \quad \text{con} \quad g_i \sim \text{Gumbel}(0,1)$$

dove $g_i = -\log(-\log U_i)$ con $U_i \sim \text{Uniform}(0,1)$ è il rumore di Gumbel. Per $\tau \to 0$, $z_i^{(\tau)}$ converge quasi certamente al campione categorico; per $\tau \to \infty$, converge a $\text{softmax}(\log \pi_i) = \pi_i$. Gumbel-Softmax è differenziabile rispetto a $\log \pi_i$ per ogni $\tau > 0$, permettendo la retropropagazione del gradiente. Il trade-off è tra varianza (bassa per τ piccolo, ma gradiente quasi zero) e bias (alto per τ grande, ma gradiente significativo): τ è trattato come iperparametro o come variabile addestrabile con annealing.

8.5 Il λ-Return e la decomposizione di Bellman multi-step

Il valore $V(h_t, z_t)$ delle variabili latenti nel world model viene addestrato tramite il λ-return, un target che interpola tra l'estimazione di Monte Carlo a orizzonte infinito (alta varianza, basso bias) e il TD(0) a un passo (bassa varianza, alto bias). Per una traiettoria immaginata di lunghezza H a partire dall'istante t, con ricompense r_{t+k} e stime del valore V_{t+k}, si definisce ricorsivamente il λ-return:

$$R_t^\lambda = r_t + \gamma \left[(1-\lambda)\, V(h_{t+1}, z_{t+1}) + \lambda\, R_{t+1}^\lambda\right]$$

con $R_{t+H}^\lambda = V(h_{t+H}, z_{t+H})$ come condizione terminale. Esplicitando la ricorrenza si ottiene la forma espansa:

$$R_t^\lambda = (1-\lambda) \sum_{n=1}^{H-1} \lambda^{n-1} G_t^{(n)} + \lambda^{H-1} G_t^{(H)}$$

dove $G_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n V(h_{t+n}, z_{t+n})$ è il return n-step. Questa rappresentazione mostra che R_t^λ è una media esponenzialmente pesata dei return n-step: λ vicino a 1 pesa i return a

lungo orizzonte (minore bias di bootstrap, maggiore varianza); λ vicino a 0 riduce al TD(1) con valore corrente come bootstrap.

Il critic V_ξ(h_t, z_t) è addestrato minimizzando la perdita di regressione sui target λ-return:

$$L_{critic}(\xi) = E_{\tau \sim \pi} [\Sigma_{t=1}^{H} \|V_\xi(h_t, z_t) - sg(R_t^\lambda)\|^2]$$

dove sg(·) indica lo stop-gradient (il target R_t^λ non propaga il gradiente a V_ξ). L'actor viene aggiornato massimizzando il valore atteso dei λ-return rispetto all'entropia della policy (per promuovere l'esplorazione):

$$L_{actor}(\psi) = E_{\tau \sim \pi_\psi} [\Sigma_{t=1}^{H} (-R_t^\lambda - \eta H[\pi_\psi(a_t \mid h_t, z_t)])], \quad \eta > 0$$

Il gradiente di L_{actor} rispetto a ψ richiede la differenziazione del λ-return rispetto alle azioni della policy. Per azioni continue, questo avviene tramite il reparametrization trick (se π_ψ è una gaussiana); per azioni discrete, tramite gradiente straight-through. In entrambi i casi, il gradiente fluisce dal valore accumulato lungo H passi di simulazione immaginata, attraverso i modelli di transizione e reward, fino ai parametri della policy.

8.6 La trasformazione symlog: fondamento statistico e connessione con l'utilità

La trasformazione symlog(x) = sign(x) log(|x| + 1) introdotta in Dreamer V3 per normalizzare le osservazioni e le ricompense ha un fondamento statistico preciso: essa è la trasformazione Box-Cox generalizzata alla distribuzione simmetrica, nel limite del parametro $\lambda \to 0$. La famiglia di trasformazioni di Box-Cox è:

$$y^{(\lambda)} = \{(x^\lambda - 1)/\lambda \ \text{ se } \lambda \neq 0, \ \log x \ \text{ se } \lambda = 0\} \ \ (\text{per } x > 0)$$

La symlog è la versione simmetrica di questa trasformazione applicata all'intera retta reale, e soddisfa le condizioni: (1) è strettamente crescente per ogni x; (2) è lineare per $|x| \ll 1$ (preserva le piccole variazioni); (3) cresce logaritmicamente per $|x| \gg 1$ (comprime le code). La funzione inversa — denominata symexp — soddisfa symexp(symlog(x)) = x e

consente di ricostruire esattamente i valori originali dalla rappresentazione trasformata.

Il vantaggio della symlog per l'addestramento dei world model è quantificabile: se la distribuzione delle ricompense ha varianza σ^2 e curtosi κ (quarto momento normalizzato), la varianza della distribuzione trasformata symlog(r) scala come $O(\log^2(\sigma))$ invece di $O(\sigma^2)$, rendendo la funzione di loss del decoder di ricompensa molto meno sensibile agli outliers ad alto valore. Questo è equivalente all'adozione di una funzione di costo pseudo-Huber:

$$L_H(r, \hat{r}) = \|symlog(r) - \hat{r}\|^2 \approx \{(r - \hat{r})^2 \ \ per\ |r - \hat{r}| \ll 1, \quad 2|r - \hat{r}| \ \ per\ |r - \hat{r}| \gg 1\}$$

La connessione con la teoria dell'utilità è la seguente: symlog(x) è la funzione di utilità di un agente con avversione crescente al rischio. L'agente si comporta come un massimizzatore di utilità attesa E[symlog(r)] rispetto a uno che massimizza E[r]: preferisce certezze moderate a scommesse ad alta varianza, un'ipotesi ragionevole per sistemi che devono operare in modo affidabile in ambienti fisici.

8.7 TD-MPC2 e World Models Latenti Efficienti

Dreamer V3 (§8.1–8.6) adotta un RSSM con stati discreti e un decoder per la ricostruzione dell'osservazione. TD-MPC2 (Hansen et al., 2024) propone un approccio alternativo che evita la ricostruzione esplicita e opera interamente nello spazio latente, raggiungendo una maggiore efficienza campionaria e scalabilità a task di manipolazione complessa.

Architettura TD-MPC2

TD-MPC2 è basato su quattro componenti. Il primo è l'encoder di osservazioni $e_\theta: O \to Z$ che mappa le osservazioni in uno spazio latente $z \in \mathbb{R}^{d_z}$. Il secondo è il dynamics model (latent transition) $d_\theta: Z \times A \to Z$ che predice lo stato latente successivo. Il terzo è il reward model $r_\theta: Z \times A \to \mathbb{R}$ che predice la ricompensa immediata. Il quarto è il value model $V_\theta: Z \to \mathbb{R}$ che stima il valore del ritorno atteso. L'obiettivo di addestramento è una combinazione di:

$$L\_TD = \| r\_\theta(z_\circledcirc, a_\circledcirc) - r_\circledcirc \|^2 + \| V\_\theta(z_{\circledcirc+1}) - y_\circledcirc \|^2$$

$$L\_consist = \| d\_\theta(z_\circledcirc, a_\circledcirc) - sg(e\_\theta(o_{\circledcirc+1})) \|^2$$

dove $y_\circledcirc$ è il λ-return (§8.5) e $sg(\cdot)$ è lo stop-gradient. A differenza di Dreamer V3, non c'è termine di ricostruzione dell'osservazione: il modello apprende rappresentazioni solo in funzione dell'utilità per la pianificazione, non per la ricostruzione dell'immagine. Questa scelta evita il problema dell'informazione irrilevante (textures di sfondo, illuminazione) che occupa capacità di modello senza contribuire alla qualità del controllo.

Pianificazione con MPPI nel World Model

TD-MPC2 adotta Model Predictive Path Integral (MPPI) come algoritmo di pianificazione nel world model latente. MPPI è una variante del model predictive control stocastico che approssima la distribuzione ottimale delle traiettorie tramite sampling importance-resampling. Data una distribuzione nominale $U = \{u_1,...,u\_H\} \sim N(\mu, \Sigma)$, MPPI campiona K traiettorie, calcola il costo per ciascuna e re-pesatura la media tramite softmax dei costi:

$$\tilde{\mu}_\circledcirc = (\Sigma_\circledcirc \exp(-J(k)/\lambda)\, u_\circledcirc(k)) \,/\, \Sigma_\circledcirc \exp(-J(k)/\lambda)$$

dove $J(k) = \Sigma_\circledcirc [-r\_\theta(z_\circledcirc(k), a_\circledcirc) + \lambda \log p(a_\circledcirc(k))/p_0(a_\circledcirc(k))]$ è il costo di informazione e λ è la temperatura. Il vantaggio di MPPI rispetto al CEM (Cross-Entropy Method) adottato in Dreamer è la coerenza con il principio di ottimalità stocastica: la distribuzione risultante è quella che minimizza il free energy $F = E[J] + \lambda\, KL(p \,\|\, p_0)$, combinando ottimizzazione del ritorno e robustezza alla divergenza dalla nominal policy.

Proposizione 8.1 (Convergenza MPPI al controllo ottimo stocastico, Williams et al. 2017).

Sia $J(\tau)$ il costo di una traiettoria nel world model. Con $K \to \infty$ campioni, la stima MPPI della distribuzione ottimale $p^*(\tau) \propto exp(-J(\tau)/\lambda)$ converge con tasso $1/\sqrt{K}$ nell'errore quadratico medio sulla distribuzione delle azioni medie $\{\tilde{\mu}_\circledcirc\}$.

Dimostrazione. La convergenza segue dalla legge dei grandi numeri applicata all'estimatore importance-weighted $E\_p[f] = E\_\{p_0\}[f(\tau) \exp(-J(\tau)/\lambda)] \,/\, E\_\{p_0\}[\exp(-J(\tau)/\lambda)]$. Con K campioni i.i.d. da p_0, la varianza dell'estimatore scala come $O(1/K)$. Il bound $O(1/\sqrt{K})$

sull'errore quadratico segue dalla disuguaglianza di Chebyshev applicata all'estimatore della media. □

Capitolo 9 — Simulatori Fisici Differenziabili: MuJoCo MJX, Genesis, IsaacLab

La simulazione differenziabile è la tecnica per cui i parametri di un simulatore fisico vengono resi dipendenti dal gradiente, in modo che algoritmi di ottimizzazione basati sul gradiente possano ottimizzare direttamente attraverso la simulazione. Questa capacità trasforma il simulatore da mero ambiente di valutazione a componente attiva del processo di apprendimento.

MuJoCo MJX è l'implementazione su acceleratori (GPU/TPU) del simulatore MuJoCo, sviluppata sfruttando JAX come backend computazionale. La scelta di JAX non è casuale: JAX supporta nativamente la differenziazione automatica di ordine arbitrario e la vectorizzazione tramite vmap per eseguire migliaia di simulazioni in parallelo sullo stesso batch GPU. MuJoCo MJX mantiene l'implementazione originale del time-stepping semi-implicito di Euler con gestione dei contatti tramite LCP softened, ma ne esegue tutto il calcolo su tensor cores, con speedup tipici di 100–1000× rispetto alla versione CPU.

9.1 Fondamenti della Simulazione Differenziabile

Un simulatore differenziabile implementa una mappa s_{t+1} = f(s_t, a_t; θ_{sim}), dove θ_{sim} sono i parametri fisici (masse, inerzie, coefficienti di attrito). La differenziabilità richiede che ∂f/∂s_t, ∂f/∂a_t, ∂f/∂θ_{sim} siano ben definiti quasi ovunque. Il problema principale è la non-differenziabilità degli eventi di contatto. Si distinguono tre approcci.

Il primo approccio è la softening dei contatti: si sostituisce la condizione di complementarità dura con una versione morbida tramite una funzione penalità differenziabile. Il secondo approccio è la differenziazione attraverso il LCP tramite la derivata implicita della soluzione rispetto ai parametri, che si ottiene dalla condizione di stazionarietà delle condizioni di complementarità:

$$\partial\lambda^* / \partial\theta = -(J\_a^\top M^{-1} J_a)^{-1} J^\top a\, M^{-1} (\partial M/\partial\theta)\, M^{-1} J_a \lambda^* + \cdots \tag{9.1}$$

dove J_a è la sottomatrice di J_c corrispondente ai contatti attivi. Il terzo approccio, implementato in Genesis (CMU, 2024), è la smoothing stocastica: si aggiunge rumore gaussiano alle configurazioni di contatto, calcolando il gradiente come valore atteso dello Jacobiano su realizzazioni disturbate.

9.2 IsaacLab: Reinforcement Learning Massivamente Parallelo

IsaacLab sfrutta la GPU sia per la simulazione fisica (tramite PhysX 5) sia per l'esecuzione della policy network, eliminando il collo di bottiglia del trasferimento dati CPU-GPU. La formula chiave che governa l'efficienza del training parallelo è: se il training richiede N campioni totali, con k ambienti in parallelo e latenza di un passo τ, il tempo totale è $T = (N/k) \cdot \tau$ + overhead. Per $N = 10^8$ campioni, $k = 10^4$, $\tau = 1$ ms si ottiene $T \approx 10^4$ s ≈ 3 ore, rispetto alle ≈ 27 ore di un training sequenziale.

9.2+ Domain Randomization: dal Trucco Empirico alla Teoria Formale

La domain randomization, introdotta da Tobin et al. (2017) per la visione artificiale e successivamente estesa al reinforcement learning fisico da OpenAI (2019) e da altri gruppi, è diventata il metodo standard per ridurre il sim-to-real gap nei sistemi robotici. Tuttavia, la sua comprensione teorica è rimasta a lungo superficiale — uno 'stratagemma' che funziona empiricamente senza una giustificazione rigorosa. Nelle sezioni seguenti si sviluppa l'interpretazione bayesiana della domain randomization e si derivano bound formali sul trasferimento sim-to-real.

9.2.1 Il Problema del Sim-to-Real Gap: Formulazione Formale

Consideriamo un sistema di controllo robotico descritto da una famiglia parametrica di MDP (Markov Decision Process):

$$M(\xi) = (S, A, T(\cdot|\cdot,\cdot;\xi), R(\cdot,\cdot;\xi), \gamma) \quad (9.2.1)$$

dove $\xi \in \Xi \subseteq \mathbb{R}^d$ è il vettore dei parametri fisici (masse, attriti, rigidità, dimensioni geometriche, parametri del modello di contatto, ecc.). Il simulatore opera con la distribuzione dei parametri P_sim(ξ), mentre il mondo reale corrisponde a una distribuzione incognita P_real(ξ). Il sim-to-real gap emerge perché P_sim ≠ P_real.

Una policy π: S → P(A) addestrata solo sul simulatore ottimizza il valore atteso rispetto a P_sim:

$$J_{sim}(\pi) = \mathbb{E}_{\{\xi \sim P_{sim}\}}[V^{\pi}(s_0;\xi)] \quad (9.2.2)$$

ma viene valutata nel mondo reale con parametri ξ_real ~ P_real:

$$J_{real}(\pi) = \mathbb{E}_{\{\xi \sim P_{real}\}}[V^{\pi}(s_0;\xi)] \quad (9.2.3)$$

Il sim-to-real gap è la differenza $|J_{sim}(\pi) - J_{real}(\pi)|$. L'obiettivo della domain randomization è addestrare π in modo che J_real(π) sia massimizzato nonostante l'incertezza in P_real.

9.2.2 Domain Randomization come Inferenza Bayesiana

La chiave per comprendere la domain randomization come regolarizzazione bayesiana è riconoscere che essa corrisponde a un'inferenza sulla distribuzione a posteriori dei parametri del mondo reale, condizionata sulle osservazioni.

Framework Bayesiano: Denotiamo con $P(\xi)$ la distribuzione a priori sui parametri fisici (encoding la nostra incertezza pre-osservazione), e con $P(D \mid \xi)$ la likelihood di osservare dati D in un sistema con parametri ξ. La distribuzione a posteriori è:

$$P(\xi \mid D) \propto P(D \mid \xi)\, P(\xi) \quad (9.2.4)$$

La domain randomization con distribuzione $P_{rand}(\xi)$ può essere interpretata come:

Caso 1 — Prior uniforme: Se $P_{rand}(\xi)$ è uniforme su un ipercubo $\Xi = [\xi_{min}, \xi_{max}]$, essa codifica una prior non informativa (massima ignoranza) sui parametri. Addestrare con $P_{rand} = U(\Xi)$ è equivalente a usare la prior di Jeffreys nell'inferenza bayesiana dei parametri.

Caso 2 — Prior gaussiana: Se $P_{rand}(\xi) = N(\mu_{nom}, \Sigma_{rand})$, essa corrisponde a una prior gaussiana centrata sui valori nominali del simulatore. La matrice di covarianza Σ_{rand} codifica l'incertezza di modellazione: elementi diagonali grandi indicano alta incertezza su quel parametro.

Caso 3 — Posterior aggiornata: Dopo aver osservato dati D dal robot reale (es. dati di rollout), è possibile aggiornare P_{rand} usando (9.2.4). Questo costituisce il sistema adaptive domain randomization o Bayesian domain randomization (BDR), dove P_{rand} viene ricalibrata iterativamente.

La connessione formale si stabilisce tramite il seguente risultato:

Proposizione (Equivalenza DR-MAP): Addestrare una policy π con domain randomization P_{rand} e ottimizzare $\mathbb{E}_{\{\xi \sim P_{rand}\}}[J(\pi; \xi)]$ è equivalente a trovare la policy che massimizza il valore atteso rispetto alla distribuzione a posteriori $P(\xi \mid D_{rand})$, dove D_{rand} è il dataset virtuale di esperienze generate dalla prior P_{rand}. In termini Bayesiani, DR realizza una stima MAP (Maximum A Posteriori) del valore rispetto alla prior P_{rand}.

$$\pi^*_{DR} = \text{argmax}_\pi \mathbb{E}_{\xi \sim P_{rand}} [J(\pi; \xi)] = \text{argmax}_\pi \mathbb{E}_{\xi \sim P(\xi | D_{rand})} [J(\pi; \xi)] \quad (9.2.5)$$

▸ *Implicazione pratica: La scelta di P_rand dovrebbe riflettere la nostra conoscenza a priori sulla distribuzione dei parametri reali. Una P_rand troppo ristretta (poca randomizzazione) porta a overfitting al simulatore; una P_rand troppo larga porta a una policy conservativa che performa subottimalmente anche nel simulatore.*

9.2.3 Bound sul Sim-to-Real Gap: Formulazione con Divergenza

Il risultato chiave è un bound esplicito sul sim-to-real gap in funzione di una misura di distanza tra P_sim e P_real. Utilizziamo due tipi di bound, basati rispettivamente sulla Divergenza di Kullback-Leibler (KL) e sulla distanza di Wasserstein (W_1).

Bound basato sulla Divergenza KL

Teorema (Sim-to-Real Gap via KL, derivato da Mansour et al. 2009): Sia V_max il valore massimo raggiungibile da qualsiasi policy in qualsiasi MDP M(ξ), e sia L_π il modulo di Lipschitz del valore della policy rispetto ai parametri ξ (ovvero: $|V^\pi(s; \xi_1) - V^\pi(s; \xi_2)| \leq L_\pi \|\xi_1 - \xi_2\|$). Allora per ogni policy π:

$$|J_{sim}(\pi) - J_{real}(\pi)| \leq L_\pi \cdot W_1(P_{sim}, P_{real}) \quad (9.2.6)$$

dove $W_1(P, Q) = \inf_{\gamma \in \Gamma(P,Q)} \mathbb{E}_{(\xi,\xi') \sim \gamma} [\|\xi - \xi'\|]$ è la distanza di Wasserstein-1 (o Earth Mover's Distance) tra le distribuzioni.

Una versione più computazionalmente trattabile usa la divergenza KL. Applicando l'inequazione di Pinsker e sfruttando la boundedness del valore:

$$|J_{sim}(\pi) - J_{real}(\pi)| \leq V_{max} \cdot \sqrt{(D_{KL}(P_{real} \| P_{sim}) / 2)} \quad (9.2.7)$$

Questo bound è valido per qualsiasi coppia di distribuzioni con supporto su Ξ e ha il vantaggio di essere stimabile anche senza un campione da P_real (tramite metodi di stima della densità o tecniche di system identification).

Bound per la Domain Randomization Ottimale

Un corollario importante riguarda la scelta ottimale di P_rand. Se si dispone di una stima P̂_real della distribuzione reale dei parametri (e.g., ottenuta da osservazioni parziali o calibrazione), la policy ottimale da addestrare è quella che minimizza il bound:

$$P^*_{rand} = \arg\min_{P_{rand}} \sup_{P_{real}: D_{KL}(P_{real} \| \hat{P}_{real}) \leq \varepsilon} \left| \mathbb{E}_{P_{rand}}[J(\pi; \xi)] - \mathbb{E}_{P_{real}}[J(\pi; \xi)] \right| \quad (9.2.8)$$

Questo è un problema minimax: si cerca la distribuzione di addestramento che minimizza il worst-case gap tra prestazione simulata e reale, sotto l'ipotesi che P_real non si discosti troppo dalla stima P̂_real. La soluzione del problema (9.2.8) fornisce la giustificazione teorica per la Adaptive Domain Randomization (ADR) di OpenAI, dove i range di randomizzazione vengono aggiustati online.

9.2.4 Domain Randomization come Regolarizzazione nel Senso di Tikhonov

C'è una seconda interpretazione della DR come regolarizzazione, più vicina alla tradizione dell'apprendimento automatico. Consideriamo il problema di ottimizzazione della policy:

$$\pi^* = \arg\max_\pi \mathbb{E}_{\xi \sim P_{sim}} [J(\pi; \xi)] \quad \text{(senza DR)} \quad (9.2.9)$$

vs.

$$\pi^*_{DR} = \arg\max_\pi \mathbb{E}_{\xi \sim P_{rand}} [J(\pi; \xi)] \quad \text{(con DR)} \quad (9.2.10)$$

Il problema (9.2.9) è equivalente a ottimizzare $J(\pi; \xi_{nom})$ — un punto singolo — il che porta facilmente a overfitting. Il problema (9.2.10) ottimizza un valore atteso su una distribuzione, il che è esattamente il ruolo di un termine di regolarizzazione nella visione di Tikhonov: l'integrazione sulla distribuzione P_rand smussa la funzione obiettivo, rendendo la policy meno sensibile alle specificità del simulatore nominale.

Formalmente, si può dimostrare che, per $P_{rand} = \mathcal{N}(\xi_{nom}, \sigma^2 I)$ con $\sigma \to 0$:

$$\mathbb{E}_{\xi \sim P_{rand}}[J(\pi; \xi)] \approx J(\pi; \xi_{nom}) + (\sigma^2/2)\, \mathrm{tr}(\nabla^2_\xi J(\pi; \xi_{nom})) + O(\sigma^4) \quad (9.2.11)$$

Il termine $\mathrm{tr}(\nabla^2_\xi J(\pi; \xi_{nom}))$ è la traccia della matrice Hessiana di J rispetto ai parametri — una misura della curvatura del valore rispetto alle variazioni di ξ. Questo termine agisce come un regolarizzatore: la domain randomization penalizza implicita-mente le policy che hanno alta curvatura rispetto ai parametri fisici, favorendo soluzioni piatte e robuste.

> ▸ *Connessione con il curriculum learning (Sezione 9.3): La domain randomization progressiva (iniziare con P_rand stretta e allargarla gradualmente) è analoga al curriculum learning: si parte da problemi facili (simulatore vicino al reale) e si aumenta gradualmente la difficoltà (maggiore randomizzazione).*

9.2.5 Stima Pratica del Sim-to-Real Gap con IsaacLab

In IsaacLab, il bound (9.2.7) può essere sfruttato praticamente nel modo seguente. Supponiamo di avere accesso a un insieme di osservazioni dal robot reale $\{\hat{\xi}_i\}_{i=1}^N$ dei parametri fisici (ottenute tramite system identification o stima online). Una stima empirica della divergenza KL è:

$$\hat{D}_{KL}(\hat{P}_{real} \| P_{sim}) = (1/N) \sum_i \log[\hat{P}_{real}(\hat{\xi}_i) / P_{sim}(\hat{\xi}_i)] \quad (9.2.12)$$

Il bound (9.2.7) fornisce quindi una garanzia quantitativa: il sim-to-real gap atteso è al massimo $V_{max} \cdot \sqrt{(\hat{D}_{KL} / 2)}$. Questo può essere usato per:

(a) Verificare se la distribuzione P_sim corrente è sufficientemente vicina a P_real prima del deploy.

(b) Determinare il range ottimale di randomizzazione: se $\hat{D}_{KL}$ è grande, si deve allargare P_rand per includere P_real nel suo supporto.

(c) Monitorare il degradation nel tempo: se P_real deriva (per usura meccanica, ecc.), $\hat{D}_{KL}$ aumenta e il bound segnala la necessità di ricalibrazione del simulatore.

> ▸ *Connessione con Capitolo 10 (Flow Matching): La stima di $D_{KL}(P_{real} \| P_{sim})$ è un problema di stima della densità, per il quale i metodi di flow matching (e normalizing flows in generale) rappresentano lo stato dell'arte. L'integrazione di IsaacLab con stimatori di densità neurali per la calibrazione adattiva del simulatore è un'area di ricerca attiva.*

9.3 Gradiente attraverso il LCP: derivata implicita e condizioni di regolarità

Durante la simulazione, ad ogni passo temporale viene risolto il Linear Complementarity Problem (LCP) (H, q) per determinare le forze di reazione λ^* ai siti di contatto attivi. La matrice $H \in \mathbb{R}^{k\times k}$ è la matrice di Delassus $H = J_a M^{-1} J_a^\top$, dove J_a è la sottomatrice di J_c relativa ai contatti attivi (dove $\varphi_n \approx 0$), e q è il vettore dei vettori di accelerazione libera proiettati sugli assi di contatto.

I parametri del simulatore θ (masse, inerzie, geometrie, coefficienti di attrito) appaiono in H e q tramite le dipendenze $H = H(q_s, \theta)$ e $q = q(q_s, \dot{q}_s, \theta)$, dove $(q_s, \dot{q}_s)$ è lo stato corrente. La soluzione $\lambda^*(\theta)$ è definita implicitamente dalla condizione di stazionarità del LCP:

$$KKT(\lambda^*, \theta) \equiv [H(\theta)\,\lambda^* + q(\theta)]_{active} = 0,\;\; \lambda^* \geq 0,\;\; [H\,\lambda^* + q]_{inactive} \geq 0$$

Differenziando la condizione di attività (la prima relazione) rispetto a θ con il teorema della funzione implicita, si ottiene:

$$H(\theta)\,\partial\lambda^*/\partial\theta + (\partial H/\partial\theta)\,\lambda^* + \partial q/\partial\theta = 0$$

che si risolve per $\partial\lambda^*/\partial\theta$:

$$\partial\lambda^*/\partial\theta = -H(\theta)^{-1}\,[(\partial H/\partial\theta)\,\lambda^* + \partial q/\partial\theta]$$

Questa formula è valida quando $H(\theta)$ è invertibile sull'insieme dei contatti attivi, ovvero quando la matrice di Delassus ristretta ai contatti attivi è non singolare. La non-singolarità è garantita dal Teorema 4.1 della monografia quando J_a ha rango pieno per righe, condizione genericamente verificata eccetto nelle configurazioni singolari (ad esempio quando tre o più contatti sono quasi-complanari con il piano di appoggio).

La formula per $\partial H/\partial\theta$ esplicita la dipendenza: $H_{ij}(\theta) = (J_a M^{-1} J_a^\top)_{ij}$ e quindi:

$$\partial H_{ij}/\partial\theta = (\partial J_a/\partial\theta)\,M^{-1}\,J_a^\top + J_a\,(\partial M^{-1}/\partial\theta)\,J_a^\top + J_a\,M^{-1}\,(\partial J_a/\partial\theta)^\top$$

dove $\partial M^{-1}/\partial\theta = -M^{-1}\,(\partial M/\partial\theta)\,M^{-1}$ per la formula di differenziazione dell'inversa. Il calcolo di $\partial J_a/\partial\theta$ richiede la differenziazione degli Jacobiani cinematici rispetto ai parametri geometrici, che è fornita dall'algoritmo di Featherstone per la dinamica ricorsiva differenziabile.

Nel caso di cambiamento del set attivo (transizione di contatto), il gradiente $\partial\lambda^*/\partial\theta$ non è definito nei punti di transizione, poiché la topologia della soluzione LCP cambia discontinuamente. In questi punti si possono adottare due strategie: (1) ignorare i punti di misura zero nella pratica dell'ottimizzazione (strategia di Todorov et al. in MuJoCo); (2) usare la regola di derivazione del valore ottimo del QP parametrico, che è valida anche nei punti di transizione nelle direzioni ammissibili.

9.4 Genesis e la simulazione differenziabile per corpi deformabili

La piattaforma Genesis (CMU, 2024) estende la simulazione differenziabile oltre i corpi rigidi, includendo fluidi, corpi elastici (FEM) e granulari (MPM — Material Point Method). Il formalismo unificatore è quello dei continui deformabili, per i quali lo stato è un campo di spostamento $u(X, t): \Omega_0 \to \mathbb{R}^3$, dove $X \in \Omega_0$ è la coordinata materiale (configurazione di riferimento). Il tensore di deformazione di Green-Lagrange è:

$$E = \tfrac{1}{2}\,(F^{\top} F - I), \quad \text{con} \quad F_{ij}(X,t) = \partial x_i/\partial X_j = \delta_{ij} + \partial u_i/\partial X_j \quad \textit{(gradiente di deformazione)}$$

La densità di energia di deformazione $W(E)$ per un materiale neo-Hookeano isotropo è:

$$W(E) = (\mu/2)(\mathrm{tr}(F^{\top}F) - 3) - \mu \log J + (\lambda/2)(\log J)^2, \quad \text{con } J = \det F$$

dove μ e λ sono i parametri di Lamé del materiale. Il primo tensore di Piola-Kirchhoff (stress nel frame di riferimento) è $P = \partial W/\partial F$. Le equazioni di Eulero-Lagrange per il continuo deformabile sono la forma debole: $\int_{\Omega_0} P : \delta F \, dV = \int_{\Omega_0} \varrho_0\, \ddot{u} \cdot \delta u \, dV$ + lavoro esterno. La discretizzazione FEM approssima u con funzioni base finite (elementi lineari triangolari o tetraedrici), e la simulazione differenziabile su GPU consiste nella

differenziazione automatica dell'intero loop di integrazione FEM rispetto ai parametri del materiale μ, λ e alla geometria iniziale Ω_0.

9.5 Domain Randomization: Fondamenti Teorici e Connessione con l'Inferenza Bayesiana

La **domain randomization** (Tobin et al. 2017) è la tecnica di addestrare una policy nel simulatore su una distribuzione di parametri fisici $p(\phi)$ — masse, inerzie, attrito, geometrie, guadagni dei motori — invece che su un singolo set di parametri nominali. L'idea intuitiva è che se la policy funziona su un intervallo abbastanza ampio di parametri simulati, funzionerà anche sul robot reale. La giustificazione teorica precisa è quella della **regolarizzazione bayesiana**.

Interpretazione Bayesiana della Domain Randomization

Sia ϕ_{real} il vettore ignoto dei parametri fisici del robot reale. La domain randomization adotta una **prior** $p(\phi)$ su ϕ, e addestra la policy a massimizzare il valore atteso rispetto a questa prior:

$$\pi^* = arg \max_{\pi} E_{\phi \sim p(\phi)}\left[V_{\phi}^{\pi}\right]$$

dove V_{ϕ}^{π} è il valore della policy π nel simulatore con parametri ϕ. Questa è esattamente la formulazione del **rischio bayesiano** in teoria della decisione: si minimizza la perdita attesa rispetto alla prior sui parametri.

Proposizione 9.1 (Domain Randomization come robustezza H_∞ bayesiana). *La policy* π^* *che massimizza* $E_{\phi}\left[V_{\phi}^{\pi}\right]$ *è anche la soluzione del problema di robustezza bayesiana:*

$$\pi^* = arg \max_{\pi} \min_{q \in Q_{\epsilon}(p)} E_{\phi \sim q}\left[V_{\phi}^{\pi}\right]$$

dove $Q_{\epsilon}(p) = \{q : D_{KL}(q \| p) \leq \epsilon\}$ *è l'intorno in KL-divergenza della prior* p*, per* $\epsilon \to 0$.

Dimostrazione. Nel limite $\epsilon \to 0$, la regione $Q_\epsilon(p)$ si restringe al punto $q = p$, e il problema di robustezza si riduce alla massimizzazione di $E_p\left[V_\phi^\pi\right]$. Per $\epsilon > 0$ finito, la formulazione è quella del distributionally robust optimization, e la soluzione corrisponde alla policy che massimizza il valore nel caso peggiore nell'intorno di KL della prior. ▫

Bound sul Sim-to-Real Gap con Domain Randomization

Teorema 9.1 (Sim-to-Real Gap con domain randomization). *Sia* $p(\phi)$ *la prior sulla domain randomization e sia* ϕ_{real} *il vettore dei parametri reali. Denotando con* $\varepsilon_{sim}(\phi) = sup_{s,a} \| p_{sim}^{\phi}(s'|s,a) - p_{real}(s'|s,a)\|_{TV}$ *l'errore di modello per parametri* ϕ*, il value gap è:*

$$V_{real}\left(\pi_E\right) - V_{real}\left(\pi^*\right) \leq H\varepsilon_{imit} + H^2 R_{max} E_{\phi\sim p}\left[\varepsilon_{sim}(\phi)\right] + H^2 R_{max}\varepsilon_{mismatch}$$

dove $\varepsilon_{mismatch} = E_{\phi\sim p}\left[\varepsilon_{sim}(\phi)\right] - \varepsilon_{sim}\left(\phi_{real}\right)$ *è il mismatch tra l'errore atteso sotto la prior e l'errore reale.*

Dimostrazione. Il bound si ottiene decomponendo il value gap in tre termini: l'errore di imitazione nel simulatore (bounded da $H\varepsilon_{imit}$), il sim-to-real gap per i parametri attesi (bounded da $H^2 R_{max} E_p\left[\varepsilon_{sim}\right]$), e il mismatch tra la distribuzione degli errori nel simulatore e l'errore al punto ϕ_{real}. Il terzo termine si annulla se ϕ_{real} è nell'intervallo di supporto della prior $p(\phi)$. ▫

La conseguenza pratica è che la domain randomization funziona (il bound è piccolo) se e solo se ϕ_{real} è nel supporto di $p(\phi)$. Questo motiva la scelta di prior ampie centrate sui valori nominali — spesso distribuzioni uniformi su intervalli $\left[\phi_{nom}(1-\delta), \phi_{nom}(1+\delta)\right]$ con $\delta \in [0.1, 0.5]$ — ma non troppo ampie da rendere il problema di training intrattabile.

9.6 MuJoCo MJX e la Differenziazione Automatica con JAX

MuJoCo MJX porta l'intero pipeline di simulazione MuJoCo su acceleratori GPU/TPU attraverso JAX, con implicazioni profonde per l'addestramento di policy neurali. Il vantaggio principale non è solo la velocità, ma la capacità di calcolare gradienti esatti attraverso la simulazione tramite autodifferenziazione di ordine arbitrario, fondamentale per algoritmi che richiedono la Hessiana della dinamica o gradienti di ordine superiore.

JAX e trasformazioni funzionali per la simulazione

JAX fornisce quattro primitive di trasformazione funzionale che MJX sfrutta sistematicamente. La prima è jit (Just-In-Time compilation) che compila l'intero loop di simulazione in XLA, eliminando l'overhead di Python e producendo codice ottimizzato per il target hardware. La seconda è vmap (vectorized map) che parallelizza N simulazioni identiche in un unico batch, con speedup quasi lineare in N fino al limite della memoria GPU. La terza è grad (automatic differentiation) che calcola il gradiente $\partial f/\partial \theta$ di qualsiasi funzione scalare f rispetto ai parametri θ tramite reverse-mode autodiff. La quarta è jacfwd/jacrev che calcola la matrice Jacobiana tramite forward- o reverse-mode, rispettivamente.

L'implementazione di $\partial \text{step}/\partial x$ e $\partial \text{step}/\partial u$ in MJX sfrutta l'approccio di differenziazione implicita per i contatti (§9.3): invece di differenziare attraverso il solver LCP (che ha gradienti discontinui), MJX approssima il contatto con un modello smoothed e differenzia attraverso questo modello. La smoothing function adottata è la Fischer-Burmeister $\psi_\varepsilon(\varphi_n, \lambda_n) = \varphi_n + \lambda_n - \sqrt{(\varphi_n^2 + \lambda_n^2 + \varepsilon^2)}$ con $\varepsilon = 10^{-3}$, che produce una approssimazione C^∞ delle condizioni di complementarità.

Addestramento policy con simulazione parallela

Il pattern tipico di addestramento con MJX + JAX è il seguente: K ambienti in parallelo tramite vmap, ciascuno integrato per H passi per produrre un batch di traiettorie di lunghezza K×H. Il gradiente della policy loss rispetto ai parametri θ della rete è calcolato tramite policy gradient (REINFORCE o PPO) o tramite differenziazione attraverso la simulazione (gradient shooting). Il secondo approccio, possibile solo grazie a MJX, produce stime del gradiente a varianza significativamente inferiore e

converge in 10–100× meno campioni, a costo di un maggior costo computazionale per gradiente.

Proposizione 9.2 (Varianza dell'estimatore policy gradient vs. differenziazione diretta).

Sia $L(\theta) = E_\tau[R(\tau)]$ il ritorno atteso. La varianza dell'estimatore REINFORCE è $\mathrm{Var}(\nabla_\theta \hat{L}_{RF}) = O(H^2 \mathrm{Var}(R) / K)$, mentre la varianza dell'estimatore per differenziazione diretta è $\mathrm{Var}(\nabla_\theta \hat{L}_{diff}) = O(H^2 \|f_{xu}\|^2 \sigma^2_{dyn} / K)$ dove σ^2_{dyn} è la varianza del rumore di modello. Per $\sigma^2_{dyn} \ll \mathrm{Var}(R)$, la differenziazione diretta è superiore.

Dimostrazione. L'estimatore REINFORCE ha varianza $O(H^2 \sigma^2_R / K)$ per la varianza del reward cumulativo su H passi. La differenziazione diretta propaga i gradienti attraverso la catena di dinamica deterministiche, con varianza dovuta solo al rumore di modello (mismatch tra simulazione e realtà). Quando il simulatore è fedele al mondo reale (piccolo σ^2_{dyn}), la varianza è dominata dall'errore di approssimazione del modello, non dalla stocasticità del reward. □

Capitolo 9.7 — World Models per Corpi Deformabili e Inferenza Causale

Raccordo con la monografia: Cap. 8 (Dreamer V3, RSSM), Cap. 9.4 (Genesis, FEM).

4.1 Rappresentazioni Latenti per Tessuti e Liquidi

I world model della Parte III (RSSM di Dreamer V3) assumono implicitamente che lo stato del mondo sia compatto e che le transizioni siano relativamente lisce. Per corpi deformabili come tessuti, impasto, o liquidi, questa assunzione fallisce: lo stato ha alta dimensionalita (il campo di spostamento u(x,t) sulla mesh) e le transizioni possono essere discontinue (piegatura, lacerazione, impatto con un liquido).

L'approccio attuale di punta e la codifica dello stato deformabile in uno spazio latente tramite una rete di auto-codifica geometrica. Sia M la mesh del corpo deformabile con N nodi; la posizione dei nodi e x in R^{3N}. Il codificatore E_phi: R^{3N} -> R^{d_z} comprime lo stato in un vettore latente z di dimensione ridotta d_z << 3N. Il decodificatore D_psi: R^{d_z} -> R^{3N} ricostruisce la mesh. Il world model impara la dinamica latente:

$$z_{t+1} = f_\theta(z_t, a_t) + \sigma_w w_t, \quad w_t \sim N(0, I)$$

Architetture recenti come DiffusionFEM (Wu et al., 2024) integrano questa struttura con il modello FEM del Cap. 9.4: il codificatore e addestrato con un termine di regolarizzazione fisico che penalizza le violazioni delle equazioni di equilibrio elastico in coordinate latenti, producendo rappresentazioni latenti fisicamente interpretabili.

4.2 Teoria della Riduzione dell'Ordine del Modello (MOR) per Corpi Elastici

La riduzione dell'ordine del modello (Model Order Reduction, MOR) per sistemi elastici ha una lunga tradizione nella meccanica computazionale. Il metodo POD (Proper Orthogonal Decomposition) costruisce una base ridotta {phi_k}_{k=1}^{r} a partire da snapshot della simulazione FEM:

$$[\Phi, \Sigma, \Psi] = \mathrm{SVD}([x_1, x_2, \ldots, x_T])$$

$$z_k = \Phi_k^T x, \quad x \approx \Phi_r z_r \quad \text{(proiezione su r modi POD)}$$

Il sistema ridotto, detto POD-Galerkin, ha equazioni del moto:

$$M_r \ddot{z} + D_r \dot{z} + K_r z = f_r(t, z)$$

dove $M_r = \Phi_r^T M \Phi_r$, $K_r = \Phi_r^T K \Phi_r$ sono le matrici di massa e rigidezza ridotte. La complessita del sistema ridotto e $O(r^3)$ invece di $O(N^3)$ per il sistema FEM completo; per tessuti con $N = 10^4$ nodi e $r = 100$ modi POD, il risparmio computazionale e di 10^6.

Proposizione S4.1 (Errore di approssimazione POD).

Sia x(t) la soluzione FEM esatta e x_r(t) = Phi_r z_r(t) la soluzione ridotta con r modi POD. L'errore di proiezione soddisfa: || x(t) - x_r(t) ||_M <= sigma_{r+1} / sigma_1 * || x ||_M, dove sigma_k e il k-esimo valore singolare della matrice di snapshot.

La connessione con il world model latente e la seguente: il codificatore E_phi della rete neurale apprende implicitamente una base POD adattata ai dati di training, ma con la flessibilita di catturare relazioni non lineari tra i modi che la riduzione POD lineare non puo rappresentare.

4.3 Inferenza Causale e World Models: Contraffattuali Robotici

I world model classici (Dreamer V3, TD-MPC2) apprendono correlazioni predittive: dato lo stato s_t e l'azione a_t, prevedono lo stato successivo s_{t+1}. Questa struttura e adeguata per la pianificazione, ma non consente di rispondere a domande contraffattuali: "se il robot non avesse eseguito l'azione a_t, qual sarebbe stato lo stato successivo?". La risposta a domande contraffattuali e fondamentale per la sicurezza (analisi degli incidenti) e per il ragionamento su causalita (capire perche un task e fallito).

Il framework del do-calculus di Pearl (2000) formalizza la distinzione tra correlazione e causalita tramite il concetto di intervento. L'effetto causale dell'azione a_t sullo stato s_{t+1} e definito come:

$$P(s_{t+1} \mid do(a_t)) \neq P(s_{t+1} \mid a_t)$$

La differenza tra le due quantita e la confusione da variabili latenti di confounding: lo stato s_t e sia causalmente precedente ad a_t (il robot sceglie l'azione in base allo stato) sia causalmente precedente a s_{t+1} (la dinamica dipende dallo stato). Il do-calculus rimuove il confounding rendendo fissa la distribuzione dello stato di partenza.

4.4 World Models Causali per la Sicurezza

L'integrazione del ragionamento causale nei world model e realizzata tramite le Causal World Models (CWM), dove il grafo causale G = (V, E) specifica le dipendenze strutturali tra variabili. Per un robot manipolatore:

$$V = \{\text{posizione robot, forza sensore, posizione oggetto, proprieta fisica oggetto}\}$$

Gli archi del grafo codificano le relazioni causali: il robot causa il movimento dell'oggetto (non viceversa), le proprieta fisiche causano la risposta alla forza. Il modello CWM addestra una rete che rispetta la struttura causale G tramite la struttura Additive Noise Model (ANM): ogni nodo e una funzione dei suoi genitori piu rumore indipendente. L'inferenza sui contraffattuali si realizza tramite l'algoritmo Pearl in tre passi (abduction, action, prediction): abduction

stima le variabili esogene latenti dall'osservazione, action interviene sul grafo, prediction propaga l'intervento attraverso le dipendenze strutturali.

Capitolo 10 — Flow Matching, Trasporto Ottimale e Simulazione Differenziabile

Il flow matching è una tecnica di apprendimento generativo che modella distribuzioni di probabilità come flussi di misure, descrivendo come una distribuzione semplice (tipicamente gaussiana) viene trasportata in una distribuzione complessa da un campo vettoriale dipendente dal tempo. A differenza della diffusione stocastica, il flow matching usa flussi deterministici, ottenendo traiettorie di generazione più brevi e stabili.

10.1 Continuous Normalizing Flows e il Teorema di Chen

Un Continuous Normalizing Flow (CNF) è definito da un campo vettoriale v_θ: ℝ^d × [0,1] → ℝ^d che trasforma la distribuzione iniziale p_0 nella distribuzione target p_1 tramite l'ODE:

$$\frac{dz(t)}{dt} = v_\theta(z(t), t), \quad z(0) \sim p_0 \quad (10.1)$$

La distribuzione marginale p_t = (Φ_t)_# p_0 evolve secondo l'equazione di continuità di Liouville:

$$\frac{\partial p_t}{\partial t} + \nabla \cdot (p_t v_\theta) = 0 \quad (10.2)$$

La variazione del log-determinante del Jacobiano lungo il flusso è data dal teorema di Liouville-Jacobi:

$$\frac{d \log |\det \partial \Phi_t / \partial z|}{dt} = tr(\partial v_\theta(z(t), t) / \partial z(t)) \quad (10.3)$$

Il calcolo del trace della Jacobiana è O(d²) in generale, ma può essere approssimato in O(d) tramite l'estimatore di Hutchinson: tr(A) ≈ E_ε[ε⊤ A ε] per ε ~ N(0, I).

10.2 Flow Matching Condizionale

Il training di un CNF tramite maximum likelihood richiede di risolvere un'ODE per ogni campione del training set, il che è computazionalmente proibitivo. Il flow matching (Lipman et al., 2022; Albergo & Vanden-Eijnden, 2023) bypassa questo problema addestrando il campo

vettoriale v_θ a imitare un campo di riferimento i cui flussi trasportano p_0 in p_1, tramite la loss di regressione:

$$L_{CFM}(\theta) = E_{t\sim U[0,1],\, x\sim p_1,\, \varepsilon\sim p_0}\left[\| v_\theta((1-t)\varepsilon + tx,\, t) - (x-\varepsilon)\|^2\right] \quad (10.4)$$

Con la scelta della traiettoria lineare z(t) = (1−t)ε + tx, il target x − ε è deterministico per ogni coppia (x, ε), eliminando la varianza aggiuntiva dell'estimatore che affligge la diffusione. In pratica, il flow matching converge più rapidamente del denoising diffusion e richiede un numero inferiore di step di inferenza (tipicamente 10–50 NFE contro i 100–1000 della diffusione DDPM).

10.3 Connessione con il Trasporto Ottimale e Chiusura del Cerchio con JEPA

La scelta ottimale del piano di trasporto γ nel senso del costo quadratico è la distanza di Wasserstein W_2:

$$W_2^2(p_0, p_1) = \inf_{\gamma\in\Gamma(p_0,p_1)} \int \|x - y\|^2 \, d\gamma(x, y) \quad (10.5)$$

Il piano ottimale γ* ha la proprietà che le traiettorie indotte siano rette — geodetiche nello spazio euclideo — ovvero ψ_t*(z|x) = (1−t)x + ty per la coppia ottimale (x, y) ~ γ*. Questa connessione, stabilita formalmente da Tong et al. (2023) nei Minibatch Optimal Transport Flows, giustifica la scelta delle traiettorie lineari come approssimazione al trasporto ottimale.

Il legame con la monografia precedente è qui diretto: la distanza di Wasserstein W_2(p_0, p_1) è precisamente la metrica introdotta in connessione con i JEPA (Joint Embedding Predictive Architectures). In quel contesto, W_2 misurava la distanza tra la distribuzione delle rappresentazioni previste e quella delle rappresentazioni reali. In questo, governa la geometria dei flussi di generazione, chiudendo un arco concettuale tra apprendimento delle rappresentazioni e generazione di traiettorie.

Il problema di identificazione di sistema come trasporto ottimale è formulato come:

$$\theta^{*} = argmin_{\theta} W_2\left(p^{sim}\theta , p^{real}\right) \quad (10.6)$$

dove p_θ^{sim} è la distribuzione delle traiettorie simulate e p^{real} quella delle traiettorie reali. La minimizzazione può essere eseguita tramite gradiente disceso approssimando W_2 con la formulazione duale di Kantorovich, o tramite flow matching: si addestra un flusso p_θ^{sim} → p^{real} e si ottimizza θ minimizzando il costo del flusso. Questo approccio, Differentiable Simulation via Wasserstein Flows, costituisce attualmente uno dei fronti più attivi della ricerca in simulazione differenziabile.

10.4 Il teorema di Brenier e la struttura del trasporto ottimale

Il problema di trasporto ottimale di Monge-Kantorovich nella formulazione di Kantorovich (1942) cerca il piano di trasporto $\gamma \in \Gamma(p_0, p_1)$ — una distribuzione congiunta su $\mathbb{R}^d \times \mathbb{R}^d$ con marginali p_0 e p_1 — che minimizza il costo atteso:

$$W_2^2(p_0, p_1) = \inf_{\gamma \in \Gamma(p_0,p_1)} \int_{\mathbb{R}^d \times \mathbb{R}^d} \|x - y\|^2 \, d\gamma(x, y)$$

Il teorema di Brenier (1991) fornisce la struttura completa del trasporto ottimale per il costo quadratico:

> **Teorema 10.1 (Brenier 1991).** *Siano p_0, $p_1 \in P_2(\mathbb{R}^d)$ con p_0 assolutamente continua rispetto alla misura di Lebesgue. Allora esiste un'unica soluzione ottimale γ^* al problema di Kantorovich con costo $c(x,y) = \|x-y\|^2$. Questa soluzione è deterministica: $\gamma^* = (Id \times T^*)_{\#} p_0$, dove $T^*: \mathbb{R}^d \to \mathbb{R}^d$ è il gradiente di una funzione convessa $T^*(x) = \nabla\varphi(x)$ con $\varphi: \mathbb{R}^d \to \mathbb{R}$ convessa. La mappa T^* è l'unico pushforward ottimale da p_0 a p_1.*
>
> *Dimostrazione.* La dimostrazione utilizza la dualità di Kantorovich: il duale del problema primale è il problema di Kantorovich-Rubinstein su coppie di funzioni di Kantorovich (f, g) con $f(x) + g(y) \le \|x-y\|^2$. Applicando la teoria delle trasformate c-convesse, si mostra che la soluzione duale (f*, g*) con g* = f*^c (la trasformata c-coniugata) fornisce, tramite le condizioni di

complementarità, che il supporto di γ^* è incluso nel sub-differenziale di f*, che è un grafico funzionale quando p_0 è assolutamente continua. La convessità di φ segue dalla c-convessità di f*.□

La conseguenza pratica del teorema di Brenier per il flow matching è fondamentale: il trasporto ottimale W_2 è realizzato da una mappa deterministica $T^* = \nabla\varphi$, e le traiettorie del flusso ottimale sono rette (geodetiche euclidee) che connettono $x_0 \sim p_0$ a $y = T^*(x_0) \sim p_1$. Questo spiega perché il flow matching con traiettorie lineari (la scelta $z(t) = (1-t)x_0 + ty$) è la migliore approssimazione pratica al trasporto ottimale: le traiettorie lineari sono le geodetiche dell'flusso ottimale di Brenier.

Il calcolo numerico della mappa di Brenier $T^* = \nabla\varphi$ si può effettuare tramite la dualità di Kantorovich: si ottimizzano le funzioni di Kantorovich tramite reti neurali (il metodo WGAN con penalizzazione del gradiente di Gulrajani et al. 2017 è essenzialmente questo), o tramite l'algoritmo di Sinkhorn per il trasporto ottimale entropico, che approssima W_2 con una versione regolarizzata.

10.5 Il ponte di Schrödinger: trasporto ottimale entropico e regolarizzazione

Il problema di Schrödinger (1932) chiede: dato un processo di diffusione browniana con probabilità di transizione $p(y|x, t)$ tra le distribuzioni p_0 e p_1, qual è la distribuzione di probabilità del processo che minimizza la divergenza KL dalla diffusione di riferimento? Questo è il problema di trasporto ottimale entropico, la cui soluzione è il ponte di Schrödinger.

Formalmente, il problema di Schrödinger è:

$$\min_{P \in M(p_0, p_1)} KL(P \,\|\, W), \quad \text{dove } W \text{ è la misura del moto browniano}$$

con $M(p_0, p_1)$ l'insieme delle misure di cammino con marginali p_0 e p_1. La soluzione P^* ha la struttura di prodotto di Schrödinger: $dP^*/dW \propto f(x_0)\, g(x_1)$, dove f e g sono le funzioni di Schrödinger determinate dalle condizioni ai bordi. Questo equivale al problema di trasporto ottimale entropico di Cuturi (2013):

$$W_{2,\varepsilon}^2(p_0, p_1) = \inf_{\gamma \in \Gamma(p_0,p_1)} \left[\int \|x-y\|^2 \, d\gamma + \varepsilon \, KL(\gamma \,\|\, p_0 \otimes p_1)\right]$$

che è la versione regolarizzata di W_2 con il termine di entropia ε KL. Per ε → 0, $W_{2,\varepsilon} \to W_2$; per ε → ∞, $W_{2,\varepsilon} \to 0$ con il piano di trasporto che collassa al prodotto indipendente $p_0 \otimes p_1$. Il grande vantaggio computazionale è che $W_{2,\varepsilon}$ è calcolabile tramite l'algoritmo di Sinkhorn in $O(n^2 / \varepsilon)$ iterazioni su n campioni, permettendo l'ottimizzazione scalabile di funzionali di trasporto ottimale nei modelli generativi.

La connessione con i modelli JEPA è la seguente. In un J-EPA (Joint Embedding Predictive Architecture), il predictor $p_\theta: Z \to Z$ apprende la rappresentazione latente dell'osservazione futura dato il contesto, minimizzando la distanza tra la rappresentazione prevista $\hat{z}_{t+k}$ e quella osservata z_{t+k}. Se la distanza nello spazio latente è la distanza di Wasserstein W_2 tra le distribuzioni delle rappresentazioni $p_\theta(\hat{z}_{t+k})$ e $p(z_{t+k})$, allora il predictor apprende implicitamente la mappa di Brenier tra queste distribuzioni. Questo stabilisce il legame concettuale profondo tra trasporto ottimale, flow matching e predizione nello spazio latente che è al cuore dell'architettura I-JEPA di LeCun.

Il teorema di Brenier (sezione 10.4) ha mostrato che il trasporto ottimale W_2 è realizzato da una mappa deterministica lungo traiettorie rettilinee. Tuttavia, questa mappa è computazionalmente difficile da calcolare esattamente, e le traiettorie rettilinee non rispettano le costrizioni fisiche dei sistemi dinamici. Il **ponte di Schrödinger** (Schrödinger 1931, riscoperto da Föllmer 1988 e Léonard 2012) offre una versione regolarizzata del trasporto ottimale che ammette soluzioni analitiche e produce traiettorie stocasticamente regolari.

Formulazione del Problema di Schrödinger

Il **problema di Schrödinger** chiede di trovare il processo stocastico P^* su $[0, 1]$ con margini prescritti $P_0 = p_0$ e $P_1 = p_1$ che sia più vicino al **processo di riferimento** Q (tipicamente il moto Browniano standard) nella KL-divergenza:

$$P^* = arg \min_{P:P_0=p_0,P_1=p_1} D_{KL}(P\|Q)$$

Questo è il trasporto ottimale entropico: invece di minimizzare il costo atteso del trasporto $E\left[\| x_0 - x_1\|^2\right]$ (che produce traiettorie deterministiche), si minimizza la KL-divergenza rispetto al moto Browniano, che produce traiettorie stocastiche con regolarità controllata dal coefficiente di diffusione.

Teorema 10.2 (Soluzione del ponte di Schrödinger, Föllmer 1988). *Siano* $p_0, p_1 \in P_2\left(R^d\right)$ *e sia* Q *il moto Browniano con covarianza* ϵI_d. *Il problema di Schrödinger ha soluzione unica* P^*, *il cui processo margine è dato da:*

$$dx_t = \nabla log \Psi\left(x_t, t\right) dt + \sqrt{\epsilon}\, dW_t$$

dove $\Psi: R^d \times [0,1] \to R_{>0}$ *è l'unica funzione che soddisfa il sistema di equazioni di Schrödinger:*

$$\Psi(x,0) = \hat{\Psi}_0(x), \quad \Psi(x,1) = \hat{\Psi}_1(x)$$

$$\partial_t \Psi = \frac{\epsilon}{2} \Delta \Psi \quad (equazione\ del\ calore\ in\ avanti)$$

$$\partial_t \hat{\Psi} = -\frac{\epsilon}{2} \Delta \hat{\Psi} \quad (equazione\ del\ calore\ all'indietro)$$

con condizioni di coupling $\Psi(x,0)\hat{\Psi}(x,0) = p_0(x)$ *e* $\Psi(x,1)\hat{\Psi}(x,1) = p_1(x)$.

Dimostrazione. La prova usa la teoria della dualità di Kantorovich in spazio di cammini. Il problema duale del problema di Schrödinger ha soluzione $\left(\varphi, \hat{\varphi}\right)$ con $\varphi + \hat{\varphi}(y) \le c_\epsilon(x,y)$ dove $c_\epsilon(x,y) = \|x - y\|^2/(2\epsilon) + \epsilon/2 \cdot log\left[(2\pi\epsilon)^d\right]$ è il costo logaritmico entropico. Identificando $\Psi(x,t) = e^{\varphi(x)/\epsilon}$ e $\hat{\Psi}(y,t) = e^{\hat{\varphi}(y)/\epsilon}$, le condizioni di dualità equivalgono al sistema di equazioni di Schrödinger. Il campo di drift $\nabla log \Psi$ è la corrente di probabilità della soluzione ottimale. ▫

Connessione con il Flow Matching e Applicazioni Robotiche

Per $\epsilon \to 0$, il ponte di Schrödinger converge al trasporto ottimale deterministico di Brenier (Theorem 10.1): le traiettorie stocastiche si concentrano sulle geodetiche deterministiche. Per $\epsilon > 0$ finito, le

traiettorie sono stocastiche e smooth, il che è computazionalmente vantaggioso: il campo di drift $\nabla log\Psi$ può essere appresso tramite **Iterative Proportional Fitting** (IPF), un algoritmo che alterna proiezioni su p_0 e p_1 nella geometria della KL-divergenza.

La connessione con il flow matching è la seguente: il campo vettoriale del flow matching condizionale $v_t\left(x|x_0, x_1\right) = \left(x_1 - x_0\right)$ produce traiettorie rettilinee che sono le geodetiche del trasporto ottimale di Brenier per $\epsilon \to 0$. Il campo vettoriale del ponte di Schrödinger $\nabla log\Psi(x,t)$ produce le geodetiche ottimali per $\epsilon > 0$, che si curvano leggermente rispetto alle rette per evitare regioni di bassa probabilità. Per le policy robotiche, questo si traduce in traiettorie di azione più regolari e fisicamente plausibili.

Proposizione 10.1 (Efficienza del ponte di Schrödinger per policy di diffusione). *Sia* $\pi_{SB}(a|o)$ *la policy generata dal ponte di Schrödinger e* $\pi_{FM}(a|o)$ *quella del flow matching con traiettorie rettilinee. Il numero di passi di discretizzazione necessari per ottenere un errore di trasporto* δ *è:*

$$N_{SB}(\delta) = O\left(\epsilon^{-1/2}\delta^{-1}\right) \quad vs \quad N_{FM}(\delta) = O\left(\delta^{-1}\right)$$

Il ponte di Schrödinger richiede $\epsilon^{-1/2}$ *passi in più per la regolarizzazione, ma produce traiettorie con curvatura media* $E\left[\| \ddot{x}_t \|\right] = O\left(\epsilon^{-1/2}\right)$ *inferiore di un fattore* $\epsilon^{1/2}$ *rispetto al flow matching.*

PARTE IV

Apprendimento per Imitazione a Basso Costo e Robotica Domestica

Capitolo 11 — ACT, Diffusion Policy e le loro Varianti

L'apprendimento per imitazione (Imitation Learning, IL) a basso costo è emerso negli ultimi anni come la strada più praticabile verso la robotica di servizio in ambienti non strutturati: invece di specificare manualmente la policy tramite ingegneria del controllo, si raccolgono dimostrazioni umane e si addestra la policy a imitarle. Il termine 'basso costo' si riferisce sia all'hardware — manipolatori di servizio reperibili commercialmente per poche migliaia di euro, come il braccio ALOHA, il Panda di Franka, o il Koch — sia all'infrastruttura di raccolta dati, che sfrutta teleoperazione a bassa latenza tramite controller da consumatore.

L'architettura ACT (Action Chunking with Transformers), sviluppata da Zhao et al. nel 2023, ha segnato un salto qualitativo in questo dominio dimostrando che un modello relativamente semplice — una variante del CVAE con encoder e decoder basati su transformer — è sufficiente per apprendere compiti manipolativi a due braccia di considerevole complessità (come aprire bottiglie, piegare magliette, staccare nastro adesivo) con un numero di dimostrazioni dell'ordine di 50–200.

11.1 Diffusion Policy: il Denoising come Policy

Diffusion Policy, proposta da Chi et al. (2023), reinterpreta il problema dell'apprendimento per imitazione come un processo generativo condizionale: dato lo stato osservativo o_t, l'azione ottimale a_t* è il campione dalla distribuzione p(a_t | o_t) che assegna probabilità massima alle azioni dimostrate. Il processo forward è la catena q(a^k | a^{k-1}) = N(a^k; √(1−β_k) a^{k-1}, β_k I) per k = 1, …, K. Il processo inverso è approssimato dalla rete neurale ε_θ(a^k, k, o_t) che predice il rumore aggiunto. L'obiettivo di training è la DDPM loss:

$$L_{DDPM}(\theta) = E_{k, a^0, \varepsilon}\left[\|\varepsilon - \varepsilon\ \theta\ (a^0 + \sigma\_k\, \varepsilon,\ k,\ o\_t)\|^2\right] \quad (11.1)$$

dove ε ~ N(0, I) è il rumore campionato, a^0 è l'azione dimostrativa e σ_k = √(Σ_{i=1}^{k} β_i) è la deviazione standard cumulata. Un vantaggio fondamentale della Diffusion Policy rispetto ai metodi a regressione diretta è la capacità di modellare distribuzioni multimodali sull'azione: se

i dimostratori eseguono lo stesso compito in modi diversi, la Diffusion Policy può catturare entrambe le modalità, mentre una gaussiana le 'media' producendo un'azione che non corrisponde a nessuna delle due.

11.2 Connessione tra Diffusion Policy e Flow Matching

Diffusion Policy e π_0 sono intimamente connesse: entrambe modellano la distribuzione delle azioni come un processo di trasformazione continua di una distribuzione gaussiana nella distribuzione target. La differenza principale è nel processo di trasformazione: la diffusione usa una SDE con coefficiente di diffusione non nullo, mentre il flow matching usa un'ODE deterministica. L'ODE probability flow di Song et al. (2021) — il membro deterministico dell'SDE di diffusione — è:

$$\frac{dx}{dt} = f(x, t) - \frac{g^2(t)}{2} \nabla\_x \log p\_t(x) \quad (11.2)$$

che ha la stessa forma dell'ODE del flow matching con campo vettoriale v_θ(x,t) = f(x,t) − (g²(t)/2) s_θ(x,t), dove s_θ(x,t) ≈ ∇_x log p_t(x) è la score function approssimata. Da questo punto di vista, il flow matching è il limite deterministico della diffusione stocastica, con traiettorie più brevi e regolari.

11.1+ Derivazione Rigorosa dell'Obiettivo DDPM come ELBO

La Diffusion Policy, nella formulazione originale di Ho et al. (2020) per la generazione di immagini adattata da Chi et al. (2023) per le azioni robotiche, ottimizza un lower bound sull'evidenza (ELBO) su una catena di Markov latente. La derivazione che segue stabilisce il legame rigoroso tra l'obiettivo di training usato nella pratica e il principio di massima verosimiglianza (maximum likelihood) rigoroso.

11.1.1 Il Modello Generativo: Catena di Markov Forward e Reverse

Sia $a^0 \in \mathbb{R}^d$ l'azione da generare (o apprendere) — per la Diffusion Policy, $a^0 \sim p\_data(a \mid o\_t)$ è l'azione dimostrativa condizionata all'osservazione o_t. Introduciamo variabili latenti $a^1, a^2, \ldots, a^K \in \mathbb{R}^d$ che interpolano tra a^0 e rumore gaussiano puro $a^K \sim N(0, I)$.

Processo forward (forward diffusion): La catena di Markov forward è:

$$q(a^1, \ldots, a^K \mid a^0) = \Pi_{k=1}^{K} q(a^k \mid a^{k-1}) \quad (11.1.1)$$

dove ciascun passo aggiunge rumore gaussiano:

$$q(a^k \mid a^{k-1}) = N(a^k; \sqrt{(1-\beta_k)}\, a^{k-1}, \beta_k I) \quad (11.1.2)$$

con schedule di rumore $\{\beta_k\}_{k=1}^{K} \subset (0,1)$. La proprietà markoviana garantisce che la marginale a qualsiasi istante k sia:

$$q(a^k \mid a^0) = N(a^k; \sqrt{\bar{\alpha}_k}\, a^0, (1-\bar{\alpha}_k) I), \quad dove\ \bar{\alpha}_k = \Pi_{i=1}^{k} (1-\beta_i) \quad (11.1.3)$$

Questa formula è cruciale: consente di campionare a^k direttamente da a^0 senza simulare l'intera catena — basta un unico campionamento da una gaussiana. Esplicitando: $a^k = \sqrt{\bar{\alpha}_k}\, a^0 + \sqrt{(1-\bar{\alpha}_k)}\, \varepsilon$, con $\varepsilon \sim N(0, I)$.

Processo reverse (denoising): Il processo reverse è anch'esso una catena di Markov, ma nella direzione opposta:

$$p\_\theta(a^0, \ldots, a^{K-1} \mid a^K) = \Pi_{k=1}^{K} p\_\theta(a^{k-1} \mid a^k) \quad (11.1.4)$$

dove ciascun passo denoising è approssimato da una gaussiana con media e varianza parametrizzate da una rete neurale:

$$p_\theta(a^{k-1} \mid a^k) = N(a^{k-1};\, \mu_\theta(a^k, k, o_t),\, \Sigma_\theta(a^k, k, o_t)) \quad (11.1.5)$$

11.1.2 La Log-Verosimiglianza e l'ELBO

L'obiettivo ideale è massimizzare la log-verosimiglianza delle azioni dimostrate:

$$\mathbb{E}_{a^0 \sim p_{data}}[\log p_\theta(a^0 \mid o_t)] \quad (11.1.6)$$

dove la verosimiglianza marginale richiede l'integrazione su tutte le variabili latenti:

$$p_\theta(a^0 \mid o_t) = \int p_\theta(a^{0:K} \mid o_t)\, da^{1:K} \quad (11.1.7)$$

Questo integrale è intrattabile. Si applica quindi la tecnica ELBO (Evidence Lower BOund) dal framework variazionale: si introduce la distribuzione approssimante $q(a^{1:K} \mid a^0)$ (il processo forward) come distribuzione variationale, e si usa la disuguaglianza di Jensen:

$$\log p_\theta(a^0 \mid o_t) \geq \mathbb{E}_{q(a^{1:K} \mid a^0)}[\log p_\theta(a^{0:K} \mid o_t) - \log q(a^{1:K} \mid a^0)] =: ELBO \quad (11.1.8)$$

L'ELBO può essere decomposto in termini interpretabili. Scriviamo:

$$ELBO = \mathbb{E}_q[\log p_\theta(a^0 \mid a^1)] - \sum_{k=2}^{K} \mathbb{E}_q[D_{KL}(q(a^{k-1} \mid a^k, a^0) \,\|\, p_\theta(a^{k-1} \mid a^k))] - D_{KL}(q(a^K \mid a^0) \,\|\, p(a^K)) \quad (11.1.9)$$

I tre termini hanno interpretazioni precise:

$L_0 = \mathbb{E}_q[\log p_\theta(a^0 \mid a^1)]$: Termine di ricostruzione — misura quanto bene il modello ricostruisce l'azione originale dall'ultimo passo di denoising. Analogo alla log-verosimiglianza in un VAE.

$L_{\text{⚜}} = \mathbb{E}_q[D_{KL}(q(a^{k-1} \mid a^k, a^0) \,\|\, p_\theta(a^{k-1} \mid a^k))]$: Termini di coerenza — misurano quanto il processo reverse si discosta dall'inversione ottimale del processo forward. La chiave è che il posteriore forward condizionato su a^0 è analitico:

$$q(a^{k-1} \mid a^k, a^0) = N(a^{k-1};\, \tilde{\mu}_{\text{⚜}}(a^k, a^0),\, \tilde{\beta}_{\text{⚜}} I) \quad (11.1.10)$$

con media:

$$\tilde{\mu}_{\text{⚜}}(a^k, a^0) = \frac{\sqrt{\bar{\alpha}_{\text{⚜}-1}}\,\beta_{\text{⚜}}}{1-\bar{\alpha}_{\text{⚜}}}\, a^0 + \frac{\sqrt{(1-\beta_{\text{⚜}})(1-\bar{\alpha}_{\text{⚜}-1})}}{1-\bar{\alpha}_{\text{⚜}}}\, a^k \quad (11.1.11)$$

e varianza $\tilde{\beta}_t = (1-\bar{\alpha}_{t-1})/(1-\bar{\alpha}_t) \cdot \beta_t$. Questo posteriore analitico è fondamentale: le divergenze KL tra gaussiane hanno forma chiusa, rendendo i termini L_t calcolabili esattamente.

L^K = DKL(q(a^K|a^0) ‖ p(a^K)): Termine di prior — misura quanto l'ultimo stato rumoroso si avvicina alla prior gaussiana $p(a^K) = N(0, I)$. Per schedule sufficientemente lungo, questo termine è praticamente zero ($\bar{\alpha}^K \approx 0$ implica $q(a^K|a^0) \approx N(0, I)$).

11.1.3 Dalla ELBO alla DDPM Loss: la Reparametrizzazione del Rumore

L'obiettivo pratico usato nel training DDPM è una semplificazione dell'ELBO. Ho et al. (2020) dimostrano che parametrizzare la rete come predittore del rumore $\varepsilon_\theta(a^k, k)$ — invece che come predittore della media $\mu_\theta(a^k, k)$ — e scegliere pesi uniformi sui termini L_t produce un obiettivo più stabile e performante:

$$L_{DDPM}(\theta) = \mathbb{E}_{k\sim U[1,K],\, a^0\sim p_{data},\, \varepsilon\sim N(0,I)} [\|\varepsilon - \varepsilon_\theta(\sqrt{\bar{\alpha}_t}\, a^0 + \sqrt{(1-\bar{\alpha}_t)}\, \varepsilon, k, o_t)\|^2] \quad (11.1.12)$$

La derivazione del legame con l'ELBO passa per la sostituzione della parametrizzazione $\mu_\theta \leftrightarrow \varepsilon_\theta$. Dalla formula (11.1.11), si ha:

$$\mu_\theta(a^k, k) = (1/\sqrt{(1-\beta_t)})\, [a^k - (\beta_t/\sqrt{(1-\bar{\alpha}_t)})\, \varepsilon_\theta(a^k, k, o_t)] \quad (11.1.13)$$

Sostituendo nell'espressione per la KL tra gaussiane e semplificando, i termini L_t diventano proporzionali a $\|\varepsilon - \varepsilon_\theta\|^2$, recuperando esattamente la DDPM loss con pesi $(\beta_t^2)/(2\sigma_t^2\, \alpha_t(1-\bar{\alpha}_t))$. Ho et al. dimostrano empiricamente che ignorare questi pesi (cioè usare peso uniforme 1 per tutti i k) migliora la qualità dei campioni, probabilmente perché enfatizza i passi con alto livello di rumore dove l'apprendimento è più informativo.

11.1.4 Variance Reduction e tecniche di campionamento efficiente

L'estimatore DDPM del gradiente ∇_θ L_DDPM ha varianza che scala come $\text{Var} \propto K \cdot \text{Var}(\varepsilon - \varepsilon_\theta)$, dove K è il numero di passi di diffusione e $\text{Var}(\varepsilon - \varepsilon_\theta)$ è la varianza residua dell'estimatore del rumore. Per modelli VLA con K = 100 passi standard, questa varianza è tipicamente un ordine

di grandezza superiore a quella degli estimatori policy gradient usati nel RL (§23.2), richiedendo batch più grandi o tecniche di variance reduction per la convergenza stabile del training. Il metodo di variance reduction più efficace per DDPM è il conditional score matching con varianza adattativa: invece di campionare t uniformemente in {1,...,K}, si usa una distribuzione non uniforme p_t $\propto \lambda(t)$, dove $\lambda(t)$ è un peso inversamente proporzionale alla varianza dell'estimatore al tempo t.

Il peso ottimale $\lambda^*(t)$ nel senso della minima varianza totale è dato dalla condizione di isotropia dell'estimatore: Var($\varepsilon - \varepsilon_\theta(a^t, t) \mid t) \cdot \lambda^*(t) = \text{cost}$. Per processi di diffusione gaussiani con schedule $\bar{\alpha}_t$, si ha Var($\varepsilon - \varepsilon_\theta \mid t$) $\propto (1 - \bar{\alpha}_t) / \bar{\alpha}_t$ (il rapporto segnale-rumore inverso), producendo $\lambda^*(t) \propto \bar{\alpha}_t / (1 - \bar{\alpha}_t) = \text{SNR}(t)$. Questo peso è esattamente il peso SNR introdotto da Hang et al. (2023) nel framework Min-SNR loss: $L_{\text{MinSNR}}(\theta) = \mathbb{E}_{t,a^0,\varepsilon}[\min(\text{SNR}(t), \gamma_{\text{clip}}) \cdot \|\varepsilon - \varepsilon_\theta\|^2]$, dove il clip superiore γ_{clip} previene pesi eccessivi ai passi con alto SNR (tipicamente t piccolo, vicino all'azione pulita) che potrebbero destabilizzare il training.

> Teorema 11.1 (Ottimalità della Min-SNR loss per la varianza del gradiente). La loss L_{MinSNR} con weight $\lambda(t) = \min(\text{SNR}(t), \gamma_{\text{clip}})$ minimizza la varianza totale dell'estimatore del gradiente $\mathbb{E}_t[\text{Var}(\nabla_\theta L_{\text{DDPM}} \mid t) \cdot \lambda(t)^2]$ soggetta al vincolo $\sum_t \lambda(t) = K$, per $\gamma_{\text{clip}} \to \infty$ (senza clipping). Con clipping γ_{clip} finito, la loss produce una riduzione della varianza rispetto al peso uniforme ($\lambda = 1$ per tutti i t) di un fattore $\sum_t \text{SNR}(t)^2 / (K \cdot \mathbb{E}_t[\text{SNR}(t)]^2) \geq 1$.
>
> Dimostrazione. Il problema di ottimizzazione dei pesi $\lambda(t)$ per la varianza minima è un problema di ottimizzazione quadratica vincolata: $\min_{\{\lambda(t)\geq 0\}} \sum_t [\text{Var}(t) \cdot \lambda(t)^2]$ s.t. $\sum_t \lambda(t) = K$, dove Var(t) è la varianza condizionale al tempo t. Le condizioni KKT danno $\lambda^*(t) = \mu / (2\,\text{Var}(t))$ con μ il moltiplicatore di Lagrange, ovvero $\lambda^*(t) \propto 1/\text{Var}(t) \propto \text{SNR}(t)$ per la relazione $\text{Var}(t) \propto 1/\text{SNR}(t)$. Il guadagno di varianza rispetto al peso uniforme segue dalla disuguaglianza di Cauchy-Schwarz: $(\sum_t 1)^2 \leq K \cdot \sum_t \text{Var}(t)/\text{Var}(t)$, riarrangiando a $\sum_t \text{Var}(t) \geq (\sum_t 1)^2/K \cdot \min_t \text{Var}(t)$. □

11.1.5 Connessione tra la diffusione e la meccanica statistica di Langevin

Il processo di diffusione DDPM è profondamente connesso alla meccanica statistica tramite il processo di Langevin overdampato. L'equazione di Langevin per una particella in un potenziale U(a) con temperatura T è $\dot{a} = -\nabla U(a) + \sqrt{(2T)}\ \xi(t)$, dove $\xi(t)$ è un rumore bianco gaussiano. La distribuzione stazionaria di questo processo è la misura di Boltzmann $p(a) \propto \exp(-U(a)/T)$. Per la generazione di azioni con Diffusion Policy, si vuole campionare dalla distribuzione dei dati p_data(a | o): il potenziale corrispondente è U(a) = −T log p_data(a | o), e la score function ∇_a log p_data(a | o) = $-\nabla U(a)/T$ è esattamente il termine che guida il processo di reverse diffusion.

Questa connessione spiega perché la score function ∇_a log p_t(a) è il quantità fondamentale da apprendere: essa è il campo di deriva del processo di Langevin associato alla distribuzione p_t(a). Diversi passi di MCMC Langevin (Stochastic Gradient Langevin Dynamics, SGLD) con score function approssimata producono campioni dalla distribuzione target — questo è il principio del Langevin Monte Carlo sampling usato come alternativa all'integrazione ODE nel campionamento da Diffusion Policy. Rispetto all'integrazione ODE, SGLD introduce rumore stocastico ad ogni passo, il che può migliorare la copertura delle mode della distribuzione (riduzione del campionamento mode-seeking) a costo di una maggiore varianza dei campioni. Il tradeoff tra qualità media dei campioni e diversità è governato dalla temperatura Langevin T_samp: T_samp = 0 corrisponde all'ottimizzazione deterministica (MAP), T_samp = T_data corrisponde al campionamento dalla distribuzione target.

Il collegamento con il framework dei world model fisici (Capitolo 8) emerge tramite la teoria della free energy variazionale. Il principio di energia libera di Friston (2010) propone che il cervello biologico minimizzi la free energy F = E_q[log q(z) − log p(o,z)] dove q(z) è la distribuzione variazionale sugli stati latenti z. Per la Diffusion Policy, il processo di reverse diffusion è esattamente la minimizzazione di una free energy variazionale rispetto alla distribuzione $q(a^{0:K} \mid a^{K}) = p_\theta(a^{0:K} \mid a^{K})$: la loss DDPM è il negativo dell'ELBO, che è il lower bound sulla log-verosimiglianza e coincide con il negativo della free energy

variazionale. Questa connessione suggerisce che la Diffusion Policy apprende implicitamente un modello di mondo interno per le azioni robotiche, analago ai world model del Capitolo 8 ma nello spazio delle azioni invece che nello spazio degli stati osservati.

> ▸ *Interpretazione Bayes-ottima: La rete $\varepsilon\_\theta(a^k, k)$ apprende la stima Bayes-ottima del rumore ε dato $a^k = \sqrt{\bar{\alpha}} a^0 + \sqrt{(1-\bar{\alpha})}\, \varepsilon$. Questo è un estimatore a minima varianza della funzione del rumore, e la sua ottimalità ha implicazioni dirette sulla scelta dello schedule — discussa nella Sezione 11.1.5.*11.2+ Score Matching e Flow Matching come Casi dello Stesso Framework

11.2.1 La Score Function e il Denoising Score Matching

La score function di una distribuzione p_t(x) è il gradiente della log-densità: s(x, t) = ∇_x log p_t(x). La score function è la quantità fondamentale che collega diffusione stocastica, flow matching e la famiglia dei modelli generativi basati su energia.

Il denoising score matching (DSM), introdotto da Vincent (2011) e formalizzato da Song & Ermon (2019), addestra una rete $s\_\theta(x, t) \approx s(x, t)$ tramite l'obiettivo:

$$L\_DSM(\theta) = \mathbb{E}\_{t, x^0, x^t \sim q(x^t|x^0)} [\| s\_\theta(x^t, t) - \nabla\_{x^t} \log q(x^t|x^0) \|^2] \quad (11.2.1)$$

Il vantaggio cruciale è che il target $\nabla\_{x^t} \log q(x^t|x^0)$ è analitico per processi forward gaussiani:

$$\nabla\_{x^t} \log q(x^t|x^0) = -\varepsilon / \sqrt{(1-\bar{\alpha})} \quad \text{se } x^t = \sqrt{\bar{\alpha}} x^0 + \sqrt{(1-\bar{\alpha})}\, \varepsilon \quad (11.2.2)$$

Il legame con la DDPM loss è immediato: $s\_\theta(x^t, t) = -\varepsilon\_\theta(x^t, t) / \sqrt{(1-\bar{\alpha}_\circledcirc)}$, e la DSM loss diventa proporzionale alla DDPM loss. Quindi la Diffusion Policy è un caso speciale di denoising score matching.

11.2.2 Il Framework Unificato: Score Matching ↔ Flow Matching

Song et al. (2021) hanno dimostrato che a qualsiasi SDE (Stochastic Differential Equation) con coefficiente di diffusione non nullo corrisponde una ODE probability flow con la stessa distribuzione marginale. La SDE di diffusione forward è:

$$da^t = f(a^t, t)\, dt + g(t)\, dw \quad (11.2.3)$$

dove w è un moto browniano standard, f(·,t) è il drift (tipicamente f(a,t) = −(1/2)β(t)a per il processo DDPM a tempo continuo) e g(t) = √β(t) è il coefficiente di diffusione.

La ODE probability flow corrispondente — che ha le stesse marginali p_t ma traiettorie deterministiche — è:

$$da^t/dt = f(a^t, t) - (g^2(t)/2)\, \nabla_{a^t} \log p_t(a^t) \quad (11.2.4)$$

L'ODE (11.2.4) è esattamente un flusso nel senso del flow matching: il campo vettoriale è v(a, t) = f(a, t) − (g²(t)/2) s(a, t), dove s(a, t) è la score function. Il flow matching condizionale (Sezione 10.2 del documento principale) corrisponde alla scelta f = 0, g = 0, con campo vettoriale target v(a, t) = (a⁰ − ε) — la direzione retta da ε a a⁰.

Teorema di Equivalenza (Song et al. 2021, Lipman et al. 2022): I seguenti tre framework sono istanze dello stesso problema di apprendimento generativo condizionato:

(i) Score matching / Denoising diffusion (DDPM): si apprende s_θ(a, t) ≈ ∇ log p_t(a) tramite L_DSM, e si campiona via integrazione della ODE (11.2.4) o della SDE (11.2.3).

(ii) Flow matching (condizionale, CFM): si apprende v_θ(a, t) ≈ v(a, t) tramite L_CFM, e si campiona via integrazione della ODE dâ/dt = v_θ(â, t), a⁰ ~ N(0, I).

(iii) Consistency models (Song et al. 2023): si apprende direttamente la mappa a_t → a_0 tramite una loss di coerenza tra istanti diversi.

I tre sono equivalenti nel senso che, se i rispettivi stimatori sono ottimali (error di approssimazione nullo), generano campioni dalla stessa distribuzione p_0 = p_data. Le differenze sono nella velocità di convergenza dell'ottimizzazione, nel numero di NFE (Neural Function

Evaluations) necessari per il campionamento, e nella varianza dell'estimatore del gradiente.

$$v_\theta(a, t) = f(a, t) - (g^2(t)/2)\, s_\theta(a, t) \longleftrightarrow \varepsilon_\theta(a, t) = -\sqrt{(1-\bar{\alpha}_t)} \cdot s_\theta(a, t) \quad (11.2.5)$$

▸ *Questa equivalenza spiega perché π_0 (OpenVLA, χ_1-π_0) si può addestrare indifferentemente con una loss di diffusione o di flow matching: i due framework ottimizzano lo stesso obiettivo con parametrizzazioni diverse. Il flow matching converge più rapidamente e usa meno NFE perché le traiettorie lineari hanno varianza dell'estimatore inferiore rispetto alle traiettorie browniane della diffusione.*

11.2.3 Analisi dello Schedule Ottimale: Legame con la Stima Bayes-Ottima

La scelta dello schedule $\{\beta_k\}_{k=1}^K$ (o $\beta(t)$ nel caso continuo) ha un impatto fondamentale sulla qualità del modello e sulla velocità di convergenza. Analizziamo questo problema dalla prospettiva dell'ottimalità Bayesiana.

Il Signal-to-Noise Ratio come Metrica

Il Signal-to-Noise Ratio (SNR) al passo k è:

$$SNR(k) = \bar{\alpha}_t / (1 - \bar{\alpha}_t) \quad (11.2.6)$$

SNR(k) interpola da SNR(0) = ∞ (segnale puro) a SNR(K) ≈ 0 (rumore puro). Lo schedule ottimale è quello che distribuisce l'apprendimento uniformemente lungo il continuum SNR, minimizzando la varianza totale dell'estimatore del gradiente.

Derivazione dello Schedule Ottimale

L'errore di stima del rumore all'istante k (errore quadratico medio del estimatore Bayes-ottimo ε_θ^*) è:

$$MMSE(k) = \mathbb{E}[\|\varepsilon - \varepsilon_\theta^*(a^k, k)\|^2] = d \cdot MMSE\_scalar(SNR(k)) \quad (11.2.7)$$

dove d è la dimensione dello spazio delle azioni e MMSE_scalar(snr) = 1/(1 + snr) è l'errore di stima Bayes-ottimo per una gaussiana

monodimensionale con SNR = snr. Questa è la formula classica per lo stimatore MMSE lineare in presenza di rumore gaussiano additivo.

Il DDPM loss totale pesato uniformemente è:

$$L\_tot = \Sigma_{k=1}^{K} \mathbb{E}[\|\varepsilon - \varepsilon\_\theta(a^k, k)\|^2] \approx K \cdot \mathbb{E}\_\{SNR\}\,[MMSE(SNR)] \quad (11.2.8)$$

Per minimizzare la varianza dell'estimatore del gradiente durante il training — che è la causa principale dell'instabilità nel training dei modelli di diffusione — si vuole che ogni passo contribuisca ugualmente alla loss totale. Questo richiede che la distribuzione del SNR indotta dallo schedule sia uniforme sul range [0, SNR_max]:

$$dSNR/dk = cost \iff \text{schedule coseno: } \bar{\alpha}_k = \cos^2(\pi k/(2K)) \quad (11.2.9)$$

Il cosine schedule di Nichol & Dhariwal (2021) è derivabile da questo criterio di uniformità del SNR. Rispetto al linear schedule originale di Ho et al. (2020), il cosine schedule evita che il SNR crolli troppo rapidamente nei primi passi (dove il segnale è ancora forte) e troppo lentamente negli ultimi passi (dove il segnale è già quasi nullo).

Ottimalità Bayes: Connessione con la Teoria dell'Informazione

C'è un risultato più profondo che collega lo schedule ottimale alla teoria dell'informazione. Song et al. (2021b) dimostrano che il DDPM loss totale è un lower bound sulla quantità di informazione mutua tra a^0 e a^K durante il processo di diffusione:

$$L\_tot \geq I(a^0; a^K) + cost \quad (11.2.10)$$

dove $I(a^0; a^K) = h(a^K) - h(a^K \mid a^0)$ è l'informazione mutua, $h(\cdot)$ è l'entropia differenziale. Per il processo gaussiano, $I(a^0; a^K) = (d/2)\log(1 + SNR(k))$.

Lo schedule ottimale nel senso della massima efficienza informativa — quello che massimizza la rate of information loss per unità di step — è equivalente allo schedule che minimizza la varianza dell'estimatore nel training. Questo stabilisce il legame fondamentale: il cosine schedule non è solo un'euristica empirica, ma è (asintoticamente, per $K \to \infty$ e $d \to \infty$) lo schedule che massimizza l'efficienza dell'apprendimento nel senso dell'informazione mutua.

$$SNR\_opt(t) = SNR\_max \cdot (1 - t) \;\rightarrow\; \bar{\alpha}\_opt(t) = SNR\_max(1-t) \,/\, (1 + SNR\_max(1-t)) \quad (11.2.11)$$

Per valori di SNR_max ≈ 1 (tipici per dati robotici normalizzati), la formula (11.2.11) produce uno schedule approssimativamente coseno — confermando l'ottimalità della scelta empirica.

> ▸ *Connessione con la Sezione 11.2 del documento principale: La probability flow ODE (11.2.4) con score function ottimale e schedule coseno produce traiettorie di generazione che minimizzano la curvatura media — rendendo l'integrazione numerica con pochi passi (low-NFE) più accurata. Questo spiega empiricamente perché la Diffusion Policy con DDIM sampler a 10-20 passi mantiene prestazioni comparabili al sampler completo a 100 passi.*

11.2.4 Riepilogo: La Gerarchia dei Modelli Generativi per le Azioni

Le relazioni derivate nelle sezioni precedenti permettono di organizzare i modelli generativi per l'apprendimento per imitazione in una gerarchia coerente:

Livello 1 — Score function: L'oggetto matematico fondamentale è la score function s(a, t) = ∇ log p_t(a). Tutti i modelli generativi discussi (DDPM, DDIM, flow matching, consistency models) sono stimatori di questa funzione, con diverse parametrizzazioni e obiettivi di training.

Livello 2 — SDE vs ODE: La scelta tra SDE (diffusione stocastica) e ODE (flow matching / probability flow) bilancia diversità dei campioni vs efficienza del campionamento. La SDE è stochasticamente più ricca (può correggere errori di integrazione), l'ODE è deterministicamente più efficiente (meno NFE).

Livello 3 — Condizionamento: Il condizionamento sull'osservazione o_t (per la Diffusion Policy) si implementa passando o_t alla rete come input aggiuntivo. Il campionamento condizionato segue automaticamente dalla stima condizionale della score function s_θ(a, t | o_t).

Livello 4 — Schedule: Lo schedule ottimale (coseno o varianti) distribuisce uniformemente il SNR lungo il processo, massimizzando

l'efficienza dell'apprendimento nel senso Bayes-ottimo e minimizzando la varianza del gradiente.

11.3 Derivazione rigorosa dell'obiettivo DDPM come ELBO su catena di Markov

La **Diffusion Policy** (Chi et al. 2023) modella la distribuzione delle azioni come il processo di denoising di un modello di diffusione. La comprensione della derivazione dell'obiettivo di training è fondamentale per connettere la Diffusion Policy al flow matching e al trasporto ottimale discussi nel Capitolo 10.

La Catena di Markov in Avanti

Il processo di diffusione in avanti $q\left(a_1, ..., a_T|a_0\right)$ aggiunge progressivamente rumore gaussiano all'azione $a_0 \sim p_{data}(a|o)$ tramite la catena di Markov:

$$q\left(a_t|a_{t-1}\right) = N\left(a_t; \sqrt{1-\beta_t}a_{t-1}, \beta_t I\right)$$

Il schedule $\{\beta_t\}_{t=1}^{T}$ (con $\beta_t \in (0,1)$ crescenti) controlla la velocità di diffusione. Definendo $\alpha_t = 1 - \beta_t$ e $\alpha^-_t = \prod_{s=1}^{t} \alpha_s$, la distribuzione marginale della catena in avanti ha la forma chiusa:

$$q\left(a_t|a_0\right) = N\left(a_t; \sqrt{\alpha^-_t}a_0, \left(1-\alpha^-_t\right)I\right)$$

Per $T\to\infty$ e schedule appropriato, $q\left(a_T|a_0\right)\approx N(0,I)$ — la distribuzione dell'azione si trasforma nel rumore gaussiano standard.

L'ELBO e la Derivazione dell'Obiettivo DDPM

Il modello generativo $p_\theta\left(a_0\right) = \int p_\theta\left(a_{0:T}\right)da_{1:T}$ è il processo di denoising in senso inverso. La log-likelihood marginale $logp_\theta\left(a_0\right)$ è intrattabile, ma ammette l'**Evidence Lower BOund** (ELBO):

$$logp_\theta\left(a_0\right) \geq E_q\left[log\frac{p_\theta\left(a_{0:T}\right)}{q\left(a_{1:T}|a_0\right)}\right] = -L_{ELBO}$$

Decomponendo la bound per ogni passo della catena di Markov, si ottiene:

$$L_{ELBO} = L_T + \sum_{t=2}^{T} L_{t-1} + L_0$$

dove:

$$L_T = D_{KL}\left(q\left(a_T|a_0\right)\|p\left(a_T\right)\right)$$

$$L_{t-1} = E_q\left[D_{KL}\left(q\left(a_{t-1}|a_t, a_0\right) \| \; p_\theta\left(a_{t-1}|a_t\right)\right)\right]$$

$$L_0 = -\, E_q\left[log p_\theta\left(a_0|a_1\right)\right]$$

Il termine L_T è costante rispetto a θ (non dipende dal modello). Il termine L_{t-1} è la distanza KL tra la posterior del processo in avanti $q\left(a_{t-1}|a_t, a_0\right)$ — che ha la forma chiusa gaussiana:

$$q\left(a_{t-1}|a_t, a_0\right) = N\left(a_{t-1}; \widetilde{\mu}_t\left(a_t, a_0\right), \widetilde{\beta}_t I\right)$$

$$\widetilde{\mu}_t\left(a_t, a_0\right) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} a_0 + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t} a_t$$

— e il modello parametrico $p_\theta\left(a_{t-1}|a_t\right)$. Se $p_\theta\left(a_{t-1}|a_t\right) = N\left(a_{t-1}; \mu_\theta\left(a_t, t\right), \widetilde{\beta}_t I\right)$, la KL-divergenza si riduce a:

$$L_{t-1} = \frac{1}{2\widetilde{\beta}_t} E_q\left[\| \, \widetilde{\mu}_t\left(a_t, a_0\right) - \mu_\theta\left(a_t, t\right)\|^2\right]$$

Ho et al. (2020) mostrano che il modello equivalente che predice il rumore $\epsilon_\theta\left(a_t, t\right) \approx \epsilon$ (invece della media) produce la loss semplificata DDPM:

$$L_{DDPM} = \sum_{t=1}^{T} E_{a_0, \epsilon}\left[\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t} a_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t\right)\|^2\right]$$

Teorema 11.1 (Equivalenza DDPM e score matching). *La loss DDPM è equivalente al denoising score matching pesato:*

$$L_{DDPM} = \sum_{t=1}^{T} \frac{1-\bar{\alpha}_t}{2} E_{a_0, a_t}\left[\| \, \nabla_{a_t} log q\left(a_t|a_0\right) - s_\theta\left(a_t, t\right)\|^2\right]$$

dove $s_\theta(a_t, t) = -\epsilon_\theta(a_t, t)/\sqrt{1-\bar{\alpha}_t}$ *è lo score estimator della distribuzione marginale* $q(a_t)$.

Dimostrazione. Si sostituisce $\nabla_{a_t} \log q(a_t|a_0) = -\left(a_t - \sqrt{\bar{\alpha}_t}a_0\right)/\left(1-\bar{\alpha}_t\right) = -\epsilon/\sqrt{1-\bar{\alpha}_t}$ nella loss di denoising score matching. La proporzionalità $\left(1-\bar{\alpha}_t\right)/2$ è il peso. La loss di score matching pesata coincide con L_{DDPM} a meno di costanti di normalizzazione. ▫

11.4 Connessione tra Diffusion Policy e Flow Matching: Unificazione

Proposizione 11.1 (Diffusion Policy come caso particolare di Flow Matching). *La Diffusion Policy con processo di denoising continuo e schedule lineare* $\beta(t) = \beta_0 + \left(\beta_1 - \beta_0\right)t$ *è equivalente al flow matching condizionale con campo vettoriale:*

$$v_t(a_t|a_0) = -\frac{a_t - \sqrt{\bar{\alpha}_t}a_0}{2\left(1-\bar{\alpha}_t\right)} \cdot \frac{d\bar{\alpha}_t}{dt}$$

La traiettoria ottimale del flow matching (con $a_0 \sim N(0, I)$ *e* $a_1 \sim p_{data}$*) corrisponde alla ODE probability flow della Diffusion Policy quando si adotta la parametrizzazione dello schedule* $\bar{\alpha}_t = 1 - t$.

Dimostrazione. Il limite continuo del processo DDPM è la SDE di Itô $da_t = -\frac{\beta(t)}{2}a_t dt + \sqrt{\beta(t)}dW_t$, che ha la ODE probability flow associata $da_t = -\frac{\beta(t)}{2}\left(a_t + \nabla_{a_t} \log q_t(a_t)\right)dt$. Sostituendo lo schedule lineare $\bar{\alpha}_t = 1 - t$ e identificando lo score con il campo vettoriale del flow matching si ottiene l'equivalenza. ▫

Questa unificazione mostra che la Diffusion Policy e il Flow Matching non sono approcci alternativi ma due istanziazioni dello stesso framework di trasporto stocastico. Il Flow Matching con traiettorie rettilinee è la versione a minima curvatura (trasporto di Brenier per $\epsilon \to 0$), la Diffusion Policy è la versione regolarizzata (ponte di Schrödinger per $\epsilon > 0$). La

scelta tra i due dipende dal regime operativo: il Flow Matching richiede meno passi di denoising (5–10 vs 100 del DDPM) ma è meno robusto al rumore nelle osservazioni.

11.5 Consistency Policy: Generazione a Singolo Step

Diffusion Policy e Flow Matching richiedono un numero di Neural Function Evaluations (NFE) non trascurabile per la generazione delle azioni: anche con DDIM a 10 passi o Flow Matching a 10 passi, la latenza di generazione è di 30–100 ms per un modello da 100M parametri — troppo lenta per il controllo a 50 Hz. Consistency Policy (Chi et al., 2024) risolve questo problema addestrando il modello a generare azioni con un singolo NFE mantenendo la qualità delle azioni vicina a quella della Diffusion Policy a N passi.

Consistency Models: fondamenti

Un Consistency Model (Song et al., 2023) apprende direttamente la funzione di consistency f_θ: $(x_\circledcirc, t) \to x_0$ che mappa lo stato rumoroso al corrispondente stato pulito, per qualsiasi $t \in [0,T]$. La proprietà fondamentale è la self-consistency: $f_\theta(f_\theta(x_\circledcirc, t), 0) = f_\theta(x_\circledcirc, t)$, ovvero applicare il modello in qualsiasi punto della traiettoria di diffusione restituisce lo stesso x_0. Questa proprietà viene imposta tramite la consistency training loss:

$$L_{CT}(\theta, \theta^-) = E[d(f_\theta(x_{\circledcirc\_n}, t_\circledcirc), f_{\{\theta^-\}}(x_{\circledcirc\_\{n-1\}}, t_{\circledcirc-1}))]$$

dove d è una metrica di distanza (tipicamente la pseudo-Huber loss o l'LPIPS per immagini), θ^- è una EMA (Exponential Moving Average) dei parametri θ, e la coppia $(t_\circledcirc, t_{\circledcirc-1})$ è campionata da una schedule di consistency. La loss forza i due punti sulla stessa traiettoria di diffusione — $(x_{\circledcirc\_n}, t_\circledcirc)$ e $(x_{\circledcirc\_\{n-1\}}, t_{\circledcirc-1})$ — a mapparsi allo stesso x_0, rendendo il modello globalmente consistente.

Consistency Policy per azioni robotiche

Consistency Policy adatta questo framework al dominio della policy robotica. Sia $a \in \mathbb{R}^{d_a}$ l'azione target e $o \in O$ l'osservazione. Il modello

$f_\theta(a_\circledcirc, t, o)$ è condizionato sull'osservazione tramite cross-attention. La generazione a singolo passo è:

$$\hat{a} = f_\theta(\varepsilon, T, o), \varepsilon \sim N(0, I)$$

Un singolo forward pass produce l'azione. Per situazioni che richiedono maggiore qualità, si può usare la refinement iterativa a K passi:

$$\hat{a}_0 = f_\theta(\varepsilon, T, o), \hat{a}_1 = f_\theta(\hat{a}_0 + \sigma_1 \varepsilon_1, t_1, o), \ldots$$

che equivale a K passi di consistency sampling — ma anche con K = 1 le prestazioni sono competitive. I risultati sperimentali riportati da Chi et al. (2024) mostrano che Consistency Policy a 1 NFE raggiunge il 94% del successo di Diffusion Policy a 100 NFE, con latenza di 5–10 ms invece di 100–300 ms, su un benchmark di 12 task manipolativi con Franka Panda.

Capitolo 12 — Regressione su Varietà di Misure e Analisi della Complessità Campionaria

La prospettiva matematica più generale sull'apprendimento per imitazione è quella della regressione su spazi di misure: la policy appresa $\pi: O \to P(A)$ è una mappa dallo spazio delle osservazioni all'insieme delle distribuzioni di probabilità sullo spazio delle azioni. Il target non è una singola azione ma una distribuzione, e il costo del training misura la distanza tra la distribuzione predetta e quella dimostrativa.

12.1 Lo Spazio delle Misure come Spazio Metrico Geodesico

Lo spazio P(A) delle distribuzioni di probabilità su A, dotato della distanza di Wasserstein W_p, forma uno spazio metrico con proprietà geometriche ricche. Per p = 2 e A = $\mathbb{R}$^d, lo spazio $(P_2(A), W_2)$ è uno spazio metrico geodesico: ogni coppia di misure è connessa da una geodetica unica. Questa struttura consente di definire nozioni di convessità, di gradiente (tramite i gradienti nello spazio di Wasserstein di Ambrosio-Gigli-Savaré) e di ottimizzazione su P(A). Come proxy computazionalmente trattabile per W_2, la Maximum Mean Discrepancy con kernel gaussiano $k(x,y) = \exp(-\|x-y\|^2/(2\sigma^2))$ è:

$$MMD^2(p\_\theta,\ p\_E) \;=\; E_{x,x'\sim p\_\theta}[k(x,x')] \;-\; 2\,E_{x\sim p\_\theta,\, y\sim p\_E}[k(x,y)] \;+\; E_{y,y'\sim p\_E}[k(y,y')] \tag{12.1}$$

12.2 Complessità Campionaria del Behavioral Cloning

> **Teorema 12.1 (Complessità campionaria del Behavioral Cloning, Ross & Bagnell 2011).**
>
> *Sia π_E la policy dell'esperto, e sia π_n la policy appresa da n dimostrazioni tramite BC che minimizza l'errore di classificazione. Con probabilità almeno $1-\delta$, il value gap tra π_E e π_n su un orizzonte H è: $V(\pi\_E) - V(\pi\_n) \le H^2 \cdot \varepsilon\_n + O(H\sqrt{(\log(1/\delta)/n)})$, dove ε_n è l'errore di classificazione empirico di π_n e $V(\pi)$ è il valore atteso della policy π.*

Dimostrazione. La dimostrazione sfrutta la decomposizione del valore tramite il lemma di performance di Kakade-Langford. Il gap $V(\pi\_E) - V(\pi\_n)$ è bounded tramite la somma degli errori di imitazione su H passi; poiché gli errori si accumulano quadraticamente con l'orizzonte — effetto compounding — si ottiene il termine H^2. La stima ε_n segue dalla disuguaglianza di McDiarmid applicata alla funzione di errore empirico, che è (1/n)-Lipschitz rispetto alla i-esima dimostrazione. □

Il termine H^2 nella stima è particolarmente insidioso per compiti a lungo orizzonte: raddoppiare H richiede quadruplicare il numero di dimostrazioni per mantenere la stessa precisione. Questo spiega perché le dimostrazioni di teleoperazione funzionano bene per compiti brevi (10–30 secondi) ma scalano male su orizzonte lungo, e motiva l'uso di tecniche come DAgger.

12.3 DAgger e la Riduzione dell'Errore di Compounding

DAgger (Ross et al., 2011) è un algoritmo interattivo che alterna training e raccolta dati per costruire iterativamente un dataset di copertura. Ad ogni iterazione i, la policy π_i corrente viene eseguita nel mondo reale, e le azioni dell'esperto π_E vengono registrate negli stati visitati da π_i — invece che negli stati dell'esperto, come nel BC standard. Il dataset aggregato $D = D_1 \cup \ldots \cup D\_i$ copre progressivamente la distribuzione di stati visitata dalla policy appresa, eliminando il distributional shift che è alla radice del compounding quadratico degli errori. La stima di complessità campionaria per DAgger ha la struttura favorevole:

$$V(\pi\_E) \; - \; V(\pi\_n) \leq H \cdot \varepsilon\_n \; + \; O(H\sqrt{(log(1/\delta)/n)}) \quad (12.2)$$

dove il termine H^2 del BC è ridotto a H. Il prezzo è la necessità di accesso interattivo all'esperto durante il training, il che può essere impraticabile in molti scenari reali. Varianti come HG-DAgger cercano di bilanciare l'efficienza del BC con la correttezza distributiva di DAgger tramite campionamento adattivo delle dimostrazioni.

12.4 GAIL: apprendimento avversariale dell'imitazione e connessione con la teoria dei giochi

L'imitation learning adversarial, introdotto da Ho e Ermon (2016), ricast il problema dell'imitazione come un gioco a due giocatori: il generatore (la policy appresa π_θ) produce traiettorie, e il discriminatore (una rete $D_\varphi: S \times A \to [0,1]$) cerca di distinguere le traiettorie della policy dalle dimostrazioni dell'esperto. La policy viene addestrata a produrre traiettorie indistinguibili da quelle dell'esperto agli occhi del discriminatore.

Formalmente, l'obiettivo del gioco GAIL è la divergenza di Jensen-Shannon tra la distribuzione degli stati-azioni visitati dalla policy π e quella dell'esperto π_E:

$$L_{GAIL}(\theta, \varphi) = E_{(s,a)\sim\pi_\theta} [\log D_\varphi(s,a)] + E_{(s,a)\sim\pi_E} [\log(1 - D_\varphi(s,a))]$$

Il discriminatore ottimale per θ fisso è $D^*(s,a) = p_E(s,a) / (p_E(s,a) + p_{\pi_\theta}(s,a))$, e con questa scelta la loss del generatore diventa:

$$-L_{GAIL}(\theta, D^*) = 2\, JSD(p_E \,\|\, p_{\pi_\theta}) - \log 4$$

dove JSD è la divergenza di Jensen-Shannon. Poiché $JSD(p \,\|\, q) \geq 0$ con uguaglianza se e solo se $p = q$, il minimo globale del gioco è raggiunto quando $p_{\pi_\theta} = p_E$, ovvero quando la policy apprende esattamente la distribuzione dell'esperto. La connessione con la teoria dei giochi è che GAIL è un gioco di Nash tra il generatore e il discriminatore: il punto di equilibrio è (θ^*, φ^*) tale che nessun giocatore ha incentivo a deviare unilateralmente.

> **Teorema 12.2 (Convergenza di GAIL, Ho e Ermon 2016).** *Sia Π la classe delle policy e D la classe dei discriminatori, entrambe sufficientemente ricche. Il punto di equilibrio di Nash del gioco GAIL soddisfa $p_{\pi_{\theta^*}} = p_{\pi_E}$, ovvero la policy apprende la distribuzione di occupazione degli stati-azioni dell'esperto. La convergenza al punto di equilibrio è garantita in un regime di apprendimento a variazione decrescente quando le policy gradient sono calcolate con i gradienti corretti del discriminatore.*
>
> *Dimostrazione.* La dimostrazione segue dall'analisi del gioco GAN di Goodfellow et al. (2014) applicata alle distribuzioni

di occupazione. La condizione di equilibrio di Nash per il discriminatore richiede $D_\varphi(s,a) = p_E(s,a)/(p_E + p_\pi)$, e sostituendo nella loss del generatore si ottiene la JSD. L'unicità dell'equilibrio ($p_\pi = p_E$) segue dalla convessità-concavità del gioco minimax e dalla separazione delle classi di funzioni. □

La differenza pratica tra GAIL e Behavioral Cloning è la riduzione quadratica del compounding error: GAIL minimizza la divergenza tra distribuzioni di occupazione (somme ponderate lungo l'intera traiettoria), non l'errore di classificazione passo per passo. La stima di complessità campionaria per GAIL ha la struttura favorevole $H \cdot \varepsilon_n$ (lineare in H) invece di $H^2 \cdot \varepsilon_n$ del BC, per lo stesso motivo per cui DAgger ha il termine H invece di H^2: la distribuzione di occupazione cattura l'effetto delle deviazioni distribuzionali lungo tutto l'orizzonte.

12.5 Model-based imitation learning e distributional shift con simulatore approssimato

Quando si dispone di un simulatore approssimato del sistema fisico — come nei modelli RSSM di Dreamer o nei simulatori differenziabili — l'imitation learning può essere condotto in parte nel simulatore e in parte nel mondo reale. Il problema del distributional shift assume allora una forma più ricca: esiste un'ulteriore fonte di shift tra la distribuzione del simulatore p_{sim} e quella del mondo reale p_{real}.

Sia $\varepsilon_{sim} = \sup_{s,a} |p_{sim}(s'|s,a) - p_{real}(s'|s,a)|_{TV}$ il total variation bound sull'errore del simulatore. Il bound sul value gap tra la policy π addestrata nel simulatore e il suo comportamento nel mondo reale è:

$$V_{real}(\pi_E) - V_{real}(\pi) \leq H\,\varepsilon_{imit} + H^2\,R_{max}\,\varepsilon_{sim}$$

dove ε_{imit} è l'errore di imitazione nel simulatore, H l'orizzonte e R_{max} il reward massimo. Il termine $H^2\,\varepsilon_{sim}$ codifica il compounding dell'errore del modello: ogni passo di simulazione introduce un errore ε_{sim} nelle transizioni, che si accumula quadraticamente su un orizzonte di H passi — lo stesso fenomeno che affligge il Behavioral Cloning per il

distributional shift della policy. Questa stima motiva l'uso del fine-tuning in dominio reale (domain adaptation) dopo il training in simulatore.

12.6 Foundation Models come Prior per l'Imitazione: GROOT e GROOT-2

Un'idea emergente nella letteratura del 2024-2025 è l'utilizzo di foundation models visivi (ViT pre-addestrati su ImageNet o LAION) come prior per la policy di imitazione: invece di addestrare la policy da zero, si parte dalla struttura di rappresentazione del foundation model e la si adatta ai dati robotici con un numero minimo di modifiche. GROOT (Dou et al., 2024) è il più avanzato rappresentante di questo paradigma.

GROOT: Goal-cOnditioned Robot Task learning

GROOT usa un ViT-L pre-addestrato con DINOv2 come backbone visivo congelato e addestra solo un piccolo transformer decoder (50M parametri) che prende in input i token visivi del backbone e produce le azioni. La policy è condizionata su un goal visivo g (immagine dello stato finale desiderato) tramite il meccanismo di attenzione cross-modale. Formalmente:

$$a_◎ = \pi\_\theta(V(o_◎), V(g), t_{exe}\_◎)$$

dove $V: O \to \mathbb{R}^{N\_patch \times d\_vis}$ è il backbone DINOv2 congelato, $t_{exe}\_◎ \in [0, T\_exec]$ è il tempo di esecuzione relativo (analogo alla coordinata di fase HZD, §3.3.2), e π_θ è il decoder policy. L'idea del goal condizionamento visivo elimina la necessità di descrivere il task in linguaggio naturale: l'utente mostra un'immagine del risultato desiderato, e il robot inferisce come raggiungerlo.

Analisi della complessità campionaria per il goal-conditioned BC

Il goal condizionamento introduce una dipendenza aggiuntiva: la policy deve modellare $p(a \mid o, g)$ invece di $p(a \mid o)$. Con N dimostrazioni e G goal distinti, la distribuzione congiunta (o, g, a) ha supporto su un product space di dimensione $2 \times \dim(O) + \dim(A)$. Il bound di generalizzazione per il goal-conditioned BC è:

$$V(\pi\_E, g) - V(\pi\_n, g) \leq H^2 (\varepsilon\_n(g) + \varepsilon\_goal\_dist)$$

dove $\varepsilon\_n(g)$ è l'errore di imitazione condizionato al goal g e $\varepsilon\_goal\_dist = W_1(p\_goal\_train, p\_goal\_test)$ è la distanza di Wasserstein-1 tra la distribuzione dei goal di training e quella di test. Per goal fuori distribuzione (immagini di sceni non viste durante il training), $\varepsilon\_goal\_dist > 0$ e introduce un termine aggiuntivo di sim-to-real goal.

PARTE V

Calcolo Neuromorfico e Robotica Event-Driven

Capitolo 12.8 — Teoria della Compressione per la Robotica: Dimensione Intrinseca e Dimostrazioni Minime

Raccordo con la monografia: Cap. 7.3 (dimensione intrinseca, Proposizione 7.3), Cap. 12 (complessita campionaria).

8.1 Il Framework Unificato

La Proposizione 7.3 della monografia stabilisce che il rango ottimale di LoRA r* >= d_int, dove d_int e la dimensione intrinseca della distribuzione degli input. Il Corollario 7.2 stabilisce un bound sul numero minimo di bit per token di azione. Questa sezione unifica questi risultati in un framework coerente che collega la geometria dello spazio delle rappresentazioni con la complessita campionaria dell'apprendimento per imitazione.

8.2 Dimensione Intrinseca della Distribuzione dei Task Robotici

La dimensione intrinseca d_int di un task robotico e la dimensione effettiva della varieta sulla quale si concentrano le dimostrazioni ottimali. Per un task di "prendere una tazza", le dimostrazioni variano in: posizione della tazza (3 DOF), orientazione della tazza (3 DOF), tipo di presa (1 DOF categorico), velocita di approccio (1 DOF). Nonostante lo spazio di configurazione del robot abbia dim 2n = 60, le dimostrazioni si concentrano su una varieta di dimensione d_int ~ 8.

La stima empirica di d_int e effettuata tramite il correlation dimension estimator di Grassberger-Procaccia applicato alle traiettorie di dimostrazione: d_int = lim_{epsilon -> 0} log C(epsilon) / log epsilon, dove C(epsilon) = (2/N^2) sum_{i < j} 1(|| z_i - z_j || < epsilon) e la fraction di coppie di punti latenti entro distanza epsilon. Per task manipolativi standard, misurazioni empiriche su Open X-Embodiment indicano d_int = 6-15 in funzione della complessita del task.

8.3 Il Numero Minimo di Dimostrazioni come Funzione della Dimensione Intrinseca

Teorema S8.1 (Complessita campionaria intrinseca per behavioral cloning).

Sia pi_E la policy esperta con distribuzione di dimostrazioni concentrata su una varieta di dimensione d_int in uno spazio di osservazione di dimensione D >> d_int. Il numero minimo di dimostrazioni n* per apprendere una policy epsilon-ottimale con probabilita 1 - delta tramite behavioral cloning su un modello con rango di LoRA r >= d_int soddisfa:

n* = Omega((d_int / epsilon^2) * log(1/delta))

mentre un modello con rango r < d_int richiede n* = Omega(D/epsilon^2) dimostrazioni (dipendente dalla dimensione dello spazio ambiente, non da quella intrinseca).

Dimostrazione. Il bound inferiore Omega(d_int/epsilon^2) segue dal framework di Rademacher complexity applicato alla classe di funzioni realizzabili da un adattamento LoRA di rango r >= d_int: la complessita di Rademacher scala come sqrt(r/n), e la condizione epsilon-ottimale richiede r/n <= epsilon^2, ovvero n >= r/epsilon^2. Il bound inferiore Omega(D/epsilon^2) per r < d_int segue dalla necessita di approssimare la funzione valore in un sottospazio di dimensione r che non riesce a catturare le direzioni rilevanti di d_int: l'errore residuo scala come (d_int - r)/D, richiedendo n >= D/(epsilon^2 (d_int - r)) dimostrazioni per compensare. []

8.4 Connessione con il Teorema di VQ-VAE e la Distorsione-Tasso

Il Teorema 7.1 (distorsione ottimale con K centri: D* ~ C_d * K^{-2/d}) e il Corollario 7.2 (numero minimo di bit per token: B* >= d_int/2 * log_2(sigma_action/epsilon)) possono essere unificati nel framework di rate-distortion theory. La funzione rate-distortion per una distribuzione gaussiana d-dimensionale con varianza sigma^2 e:

$$R(D) = d/2 * \log(\sigma^2 / D)$$

dove R e il rate in bit e D e la distorsione (errore quadratico medio). Sostituendo la distorsione target D = epsilon^2 e la varianza sigma^2 = sigma_action^2, si ottiene R = d_int/2 * log_2(sigma_action^2 / epsilon^2) = d_int * log_2(sigma_action/epsilon), che coincide esattamente con il Corollario 7.2. Questo stabilisce che il numero minimo di bit per token di azione e esattamente la capacita minima di canale necessaria per trasmettere l'informazione del task con distorsione epsilon -- un risultato di Shannon che si traduce direttamente in requisiti di training per i modelli VLA.

Capitolo 12.7 — Manipolazione Bimanuale a Basso Costo e Augmentazione dei Dati in SE(3)

Raccordo con la monografia: Cap. 11–12 (ACT, Diffusion Policy, complessita campionaria).

5.1 ALOHA 2 e il Design dell'Hardware Bimanuale Open-Source

ALOHA (A Low-cost Open-source Hardware System for Bimanual Teleoperation, Zhao et al. 2023) e ALOHA 2 (2024) sono sistemi di teleoperazione bimanuale basati su bracci Dynamixel low-cost (< 20.000 USD coppia) che hanno democratizzato la raccolta di dati per manipolazione a due braccia. Il design e replicabile: due bracci leader (azionati dall'operatore) e due bracci follower (il robot) connessi da telemetria a bassa latenza.

La complessita campionaria della manipolazione bimanuale differisce qualitativamente da quella del manipolatore singolo per il coordinamento temporale richiesto. Sia a^L e a^R le azioni del braccio sinistro e destro. La policy bimanuale pi: O -> P(A^L x A^R) deve modellare la distribuzione congiunta, che non e fattorizzabile come prodotto di distribuzioni marginali:

$$P(a^L, a^R \mid o) \neq P(a^L \mid o) * P(a^R \mid o)$$

Il termine di correlazione C(a^L, a^R | o) = P(a^L, a^R | o) / [P(a^L | o) * P(a^R | o)] misura la dipendenza tra le due braccia. Per task di coordinamento preciso (piegare una maglietta, aprire un vasetto), C >> 1 nella regione di azioni ottimali: le braccia devono sincronizzarsi esattamente.

5.2 Complessita Campionaria della Manipolazione Bimanuale: Bound

Teorema S5.1 (Complessita campionaria bimanuale).

Sia pi_E^biman la policy esperta bimanuale e pi_n la policy appresa da n dimostrazioni tramite BC su architettura con head condivisa (Shared-Head Bimanual BC). Il value gap soddisfa:

$$V(\pi_{E}^{biman}) - V(\pi_{n}^{biman}) \leq H^2 * \varepsilon_n + H^2 * I(a^L; a^R \mid o) / n$$

dove I(a^L; a^R | o) e l'informazione mutua condizionata tra le azioni dei due bracci. Il secondo termine e addizionale rispetto al caso mono-braccio (Teorema 12.1) e scala con la correlazione tra le braccia.

Dimostrazione. La decomposizione della KL-divergenza tra policy bimanuale esperta e appresa si scrive come KL(P_E || P_n) = KL_L + KL_R + I(a^L; a^R | o), dove i primi due termini corrispondono agli errori individuali dei due bracci e il terzo al termine di coordinamento. Applicando il lemma di performance di Kakade-Langford separatamente ai tre termini si ottiene il bound. []

Il corollario pratico e che per task con alta coordinazione (I(a^L; a^R | o) ~ 2 bit su task di manipolazione di oggetti rigidi), il numero di dimostrazioni necessarie e 2^2 = 4x superiore a quello del caso mono-braccio.

5.3 Augmentazione dei Dati in SE(3)

La data augmentation per le dimostrazioni robotiche e tecnicamente piu complessa dell'augmentation per immagini: una dimostrazione e una traiettoria (o_{1:T}, a_{1:T}) nello spazio stato-azione congiunto, e le trasformazioni devono essere coerenti lungo tutta la traiettoria e rispettare la geometria del robot e della scena.

L'augmentazione SE(3) consiste nell'applicare una trasformazione rigida g in SE(3) all'intera scena (robot + oggetti + riferimento del task) e trasformare di conseguenza le azioni. Per una trasformazione g = (R, t) in SE(3):

I'_t = Render(R * scene_t + t) (immagine trasformata)

a'_t = R * a_t (azione trasformata: componente rotazionale)

p'_t = R * p_t + t (posizione trasformata: end-effector)

La policy appresa su dati augmentati con augmentazione SE(3) e equivariante per costruzione (Proposizione 6.1): P(a | o) = P(R*a | Render(R*o)) per ogni R in SO(3). Il guadagno in complessita campionaria e esattamente il fattore |G| della Proposizione 6.1, dove G e il gruppo di simmetria campionato durante l'augmentazione.

5.4 Implementazione Efficiente dell'Augmentazione SE(3)

L'implementazione dell'augmentazione SE(3) richiede un renderer differenziabile per produrre immagini coerenti delle scene trasformate. Due approcci sono disponibili: il rendering basato su rasterizzazione differenziabile (Pytorch3D, nvdiffrast) e il rendering basato su NeRF o Gaussian Splatting. Il secondo approccio e piu costoso ma produce immagini piu fotorealistiche, riducendo il gap tra dati augmentati e osservazioni reali.

Per dataset di piccole dimensioni (< 200 dimostrazioni), l'augmentazione SE(3) con un gruppo ciclico di 8 rotazioni planar (G = Z_8) riduce l'errore del Behavioral Cloning del 20-35% su task manipolativi standard, a costo di un aumento dell'8x del tempo di training.

Capitolo 13 — Architetture Neuromorfe: Loihi 2 e BrainScaleS-2

Il calcolo neuromorfico è un paradigma computazionale ispirato alla neurobiologia nel quale l'elaborazione dell'informazione avviene attraverso reti di neuroni artificiali che comunicano tramite eventi discreti — gli spike — propagati asincronamente. A differenza delle GPU, il cui modello computazionale è quello del parallelismo massivo sincrono (SIMD), i chip neuromorfi come Intel Loihi 2 e BrainScaleS-2 implementano un modello event-driven: l'elaborazione avviene soltanto quando uno spike viene ricevuto, e i neuroni che non ricevono input non consumano energia. Questa efficienza energetica event-driven rende l'hardware neuromorfico particolarmente adatto alle applicazioni robotiche a bassa latenza e bassa potenza.

Intel Loihi 2, presentato nel 2021, contiene 1 milione di neuroni artificiali programmabili distribuiti su 128 neuro-core connessi da un network-on-chip a topologia mesh. Ogni neuro-core implementa 1024 neuroni e gestisce la loro dinamica tramite hardware dedicato ad aggiornamento event-driven. La latenza di risposta a uno spike in ingresso è inferiore a 1 μs — tre ordini di grandezza al di sotto della latenza tipica di una GPU per un forward pass di rete neurale. La potenza dissipata per singola inferenza è dell'ordine dei pJ (picojoule), contro i nJ–μJ di una GPU.

BrainScaleS-2, sviluppato dall'Università di Heidelberg nell'ambito del programma europeo Human Brain Project, adotta un approccio radicalmente diverso: è un chip analog mixed-signal in cui la dinamica neuronale è implementata direttamente dalla fisica dei circuiti analogici. Questo consente di simulare la dinamica di neuroni LIF con un'accelerazione di circa 1000× rispetto al tempo biologico reale: un secondo di attività neuronale simulata corrisponde a 1 ms di tempo fisico. Tale accelerazione rende BrainScaleS-2 ideale per il training di reti spiking tramite simulazione accelerata.

13.1 Il Modello Leaky Integrate-and-Fire

Il modello di neurone più adottato nel calcolo neuromorfico è il Leaky Integrate-and-Fire (LIF). Il potenziale di membrana u(t) evolve secondo:

$$\tau_m \frac{du(t)}{dt} = -(u(t) - u_{rest}) + R_m I_{syn}(t) \quad (13.1)$$

dove τ_m = R_m C_m è la costante di tempo della membrana, u_{rest} è il potenziale di riposo, R_m è la resistenza di membrana e I_{syn}(t) è la corrente sinaptica. Quando u(t) supera la soglia ϑ, il neurone emette uno spike e il potenziale viene resettato a u_{reset} per un periodo refrattario τ_{ref}:

$$se\ u(t) \geq \vartheta : spike,\ poi\ u(t^+) = u_{reset}\ per\ t \in [t_{spike},\ t_{spike} + \tau_{ref}] \quad (13.2)$$

La corrente sinaptica I_{syn}(t) = Σ_j Σ_{t_j ∈ F_j} w_{ji} ε(t − t_j) è la somma delle contribuzioni di tutti i neuroni j pre-sinaptici, con w_{ji} il peso sinaptico ed ε(t) = e^{−t/τ_{syn}} per t ≥ 0 il kernel sinaptico esponenziale.

13.2 Il Modello Spike Response (SRM) e la sua Struttura Lineare

Il Spike Response Model (SRM), introdotto da Gerstner e Kistler, esprime il potenziale di membrana come convoluzione degli input sinaptici con kernel dipendenti dai tempi di spike passati:

$$u(t) = \sum_j \sum_{t^f_j} w_{ji}\, \varepsilon(t - t^f_j) + \sum_{t^f_i} \eta(t - t^f_i) \quad (13.3)$$

dove ε(t − t_j^f) è il kernel EPSP e η(t − t_i^f) è il kernel di after-potential. Questa decomposizione ha una struttura lineare rispetto ai kernel: il potenziale è la sovrapposizione di contributi di forme note traslate ai tempi di spike.

> **Proposizione 13.1 (Equivalenza LIF-SRM).**
>
> *Il modello LIF con corrente sinaptica a un esponenziale è un caso particolare del SRM con kernel ε(t) = (1/τ_m − 1/τ_{syn})^{-1} (e^{−t/τ_m} − e^{−t/τ_{syn}}) per t ≥ 0 (kernel alpha, valido per τ_m ≠ τ_{syn}) e kernel*

> *$\eta(t) = (u_{reset} - \vartheta)\, e^{-t/\tau_m}$ per $t \geq 0$. La condizione di firing corrisponde a $u(t) \geq \vartheta$ nel SRM equivalente.*

Dimostrazione. Sostituendo le espressioni dei kernel nel SRM e verificando che la dinamica del potenziale soddisfa l'equazione differenziale (13.1) con la condizione di reset (13.2), il risultato segue per unicità della soluzione del sistema di equazioni differenziali con condizioni iniziali date dal reset. □

13.3 Il modello LIF adattativo (AdEx) e la corrente di adattamento

Il modello Adaptive Exponential Integrate-and-Fire (AdEx), proposto da Brette e Gerstner nel 2005, estende il LIF con un termine esponenziale di soglia e una variabile di adattamento w:

$$C_m\, du/dt = -g_L(u - E_L) + g_L\, \Delta_T \exp((u - V_T)/\Delta_T) + I_{syn} - w$$

$$\tau_w\, dw/dt = a(u - E_L) - w$$

con la condizione di spike e reset: se u ≥ V_peak, allora u ← u_r e w ← w + b. In queste equazioni, g_L è la conduttanza di perdita, E_L il potenziale di riposo, V_T la soglia di spike, Δ_T la sharpness del trigger esponenziale, τ_w la costante di tempo della corrente di adattamento, a il parametro di accoppiamento sotto-soglia e b l'incremento di w per spike. Il termine esponenziale g_L Δ_T exp((u − V_T)/Δ_T) modella la corrente di sodio voltaggio-dipendente responsabile dell'instabilità di soglia; la variabile w modella le correnti di potassio lente.

Il comportamento del modello AdEx per input costante I è caratterizzato dal piano di fase (u, w). Per a < g_L (sub-threshold adaptation), il punto fisso (u*, w*) soddisfa g_L(u* − E_L) = w* e a(u* − E_L) = w*, quindi w* = a(u* − E_L) e u* = E_L + w*/g_L. Il ciclo di firing periodico con adattamento ha frequenza istantanea che decresce come:

$$f(t) = f_\infty + (f_0 - f_\infty)\exp(-t/\tau_{adapt}), \quad \tau_{adapt} = \tau_w / (1 - a/g_L)$$

per t dopo l'inizio dello stimolo, dove f_0 è la frequenza iniziale e f_∞ è la frequenza asintotica. Questa dinamica di adattamento è biologicamente

plausibile e riproducibile sperimentalmente nelle cellule piramidali corticali con $\tau_{adapt} \approx 100–500$ ms.

Per l'implementazione su chip neuromorfico Loihi 2, il modello AdEx discretizzato con passo dt è:

$$u_{t+1} = u_t + dt/C_m \left[-g_L(u_t - E_L) + g_L \Delta_T \exp((u_t - V_T)/\Delta_T) + I_t - w_t\right]$$

$$w_{t+1} = w_t + dt/\tau_w \left[a(u_t - E_L) - w_t\right]$$

La non-linearità esponenziale introduce una difficoltà: la funzione $\exp((u - V_T)/\Delta_T)$ non è supportata nativamente nell'hardware integer di Loihi 2. L'approssimazione lineare a tratti (PLU — Piecewise Linear Unit) con $k \geq 4$ segmenti approssima l'esponenziale con errore relativo inferiore all'1% nella regione $u \in [V_T - 2\Delta_T, V_T + 2\Delta_T]$, coprendo il range fisiologico dei potenziali di membrana sub-soglia.

Capitolo 14 — Matematica delle Reti a Picchi: Codifica Temporale e Approssimazione di Poisson

La questione della codifica dell'informazione nelle reti a picchi è uno dei problemi fondamentali del calcolo neuromorfico, e la sua risposta determina la scelta del regime operativo dell'hardware e degli algoritmi di apprendimento. Due paradigmi principali si contrappongono: la codifica per frequenza (rate coding), in cui l'informazione è trasmessa dalla frequenza di firing del neurone su una finestra temporale, e la codifica temporale (temporal coding), in cui l'informazione è portata dai tempi esatti degli spike.

14.1 Codifica per Frequenza e Approssimazione di Poisson

Nel regime di codifica per frequenza, i treni di spike sono approssimabili come processi di Poisson omogenei: la probabilità di emettere uno spike nell'intervallo [t, t+dt] è r_i(t) dt. Un processo di Poisson con rate r produce in [0, T] un numero N ~ Poisson(rT) con E[N] = rT e Var(N) = rT, ovvero CV = 1/√(rT) — approssimativamente unitario per finestre brevi, come osservato nei neuroni biologici. Per una rete con n_{pre} neuroni pre-sinaptici con rate r_{pre,j} e pesi w_{ji}, la frequenza del neurone post-sinaptico nell'approssimazione lineare è:

$$r_{post,i} = \sigma(\sum_j w_{ji}\, r_{pre,j} - \vartheta/\tau_m) \quad (14.1)$$

Questa equazione ha la stessa struttura di una rete neurale artificiale con funzione di attivazione σ, ovvero la mean-field approximation per reti spiking. Si tratta del regime in cui i risultati della teoria delle reti neurali classiche possono essere importati nel dominio spiking.

14.2 Codifica Temporale e Spike Timing Dependent Plasticity

Il meccanismo biologico di apprendimento temporale è la Spike Timing Dependent Plasticity (STDP): la sinapsi tra il neurone pre-sinaptico j e quello post-sinaptico i viene potenziata se lo spike di j precede quello di i (Long-Term Potentiation, LTP), e indebolita nel caso contrario (Long-Term Depression, LTD). La regola STDP classica è:

$$\Delta w_{ji} = f(\Delta t) = \{A_+ e^{-\Delta t/\tau_+} \; se \; \Delta t > 0, \; A_- e^{\Delta t/\tau_-} \; se \; \Delta t < 0\} \quad (14.2)$$

dove Δt = t_{post} − t_{pre}, $A_+ > 0$ e $A_- < 0$ sono le ampiezze di LTP e LTD, e τ_+, $\tau_- \approx 20$ ms le costanti di tempo. La curva f(Δt) è la curva STDP: la sua forma asimmetrica impone una dipendenza causale — soltanto le sinapsi le cui attivazioni precedono il firing post-sinaptico vengono potenziate. La connessione tra STDP e la Hebbian learning classica ($w_{ij} \propto \langle x_i x_j \rangle$) si stabilisce nel regime di basse frequenze: integrando la regola STDP sulla distribuzione temporale degli spike, si recupera una forma di apprendimento hebbiano con normalizzazione, analoga alla Independent Component Analysis (ICA).

14.3 Gradiente Surrogate e Backpropagation nelle Reti Spiking

Il principale ostacolo all'addestramento delle reti spiking tramite backpropagation è la non-differenziabilità della funzione di firing: la funzione di Heaviside H(u − ϑ) che governa l'emissione di uno spike ha derivata nulla ovunque tranne in u = ϑ, dove è infinita. La soluzione adottata nella maggioranza degli algoritmi moderni è il gradiente surrogate: la derivata della funzione di firing durante il backward pass viene sostituita con la derivata di una funzione morbida h_surr(u) che approssima H(u − ϑ). La scelta più comune è:

$$h'_{surr}(u) = \frac{1}{(\beta|u - \vartheta| + 1)^2} \quad (14.3)$$

Con questa sostituzione, le equazioni di backpropagation attraverso il tempo (BPTT) si applicano alle reti spiking con la stessa struttura delle LSTM e delle RNN convenzionali. La giustificazione teorica del gradiente surrogate è la sua interpretazione come estimatore del gradiente della funzione di valore atteso ∂E[s]/∂u, dove s = H(u − ϑ) è lo spike e l'aspettazione è rispetto alle fluttuazioni stocastiche del potenziale.

14.4 Backpropagation Through Time per reti LIF: derivazione esplicita

Una rete spiking a L layer con dinamica LIF discreta ha, per il layer l e il neurone i al tempo t, le variabili di stato: potenziale di membrana u_i^l(t) e treno di spike s_i^l(t) ∈ {0,1}. Le equazioni di aggiornamento sono:

$$u_i^l(t+1) = \beta\, u_i^l(t)\,(1 - s_i^l(t)) + \Sigma_j\, w_{ij}^l\, s_j^{l-1}(t+1)$$

$$s_i^l(t) = H(u_i^l(t) - \vartheta)$$

dove β = exp(−dt/τ_m) è il fattore di decadimento del potenziale, w_{ij}^l è il peso sinaptico e H è la funzione di Heaviside. Il termine (1 − s_i^l(t)) modella il reset: quando il neurone spikea (s = 1), il potenziale viene resettato sottraendo la soglia ϑ (reset per sottrazione). La loss L dipende dagli spike di output s_i^L(T) all'istante T finale.

Il gradiente ∂L/∂w_{ij}^l si calcola tramite la regola della catena attraverso il tempo (BPTT). Definendo il gradiente del potenziale δu_i^l(t) = ∂L/∂u_i^l(t), la retropropagazione segue le equazioni:

$$\delta u_i^l(t) = \beta\, \delta u_i^l(t+1)\,(1 - s_i^l(t)) - \beta\, u_i^l(t+1)\, \delta s_i^l(t) + \Sigma_j\, w_{ji}^{l+1}\, \delta u_j^{l+1}(t)\, h'_{surr}(u_j^{l+1}(t))$$

dove δs_i^l(t) = ∂L/∂s_i^l(t) e h'_{surr}(u) è il gradiente surrogate che sostituisce la derivata delta della funzione di Heaviside. Il gradiente del peso è:

$$\partial L/\partial w_{ij}^l = \Sigma_t\, \delta u_i^l(t+1)\, s_j^{l-1}(t+1)$$

che ha la struttura di correlazione tra il segnale di errore retropropagato e gli spike pre-sinaptici — formalmente analogo alla regola hebbiana di STDP, ma calcolato tramite retropropagazione globale invece di plasticità locale. Questa connessione è profonda: in un certo regime, BPTT e STDP convergono sulla stessa regola di aggiornamento, giustificando la biologia della STDP come approssimazione locale dell'algoritmo di apprendimento globalmente ottimo.

14.5 Complessità VC delle reti spiking e confronto con le reti analogiche

Un risultato fondamentale sulla capacità delle reti spiking è che la dimensione VC di una rete a T passi temporali con N neuroni LIF e sinapsi W è $\Theta(W\ T \log T)$, confrontata con la VC dimension di una rete neurale analogica con la stessa architettura (pesi reali, attivazione ReLU) che è $\Theta(W^2\ T)$ o $\Theta(W \log W)$ (a seconda del regime). Quindi per T piccolo (sequenze brevi), le reti spiking hanno capacità comparabile alle analogiche; per T grande, la complessità cresce solo logaritmicamente con T nelle reti spiking, poiché ogni neurone contribuisce al massimo O(log T) bit di informazione tramite il suo treno di spike.

> **Proposizione 14.1 (Bound VC per reti LIF, Maass 1997).** *Una rete spiking di tipo LIF con N neuroni, W sinapsi, e T passi temporali ha dimensione VC $\Omega(W\ T)$ e $O(W\ T \log(N\ T\ W))$, dove le costanti dipendono dalla granularità temporale.*
>
> *Dimostrazione.* Il lower bound segue dalla costruzione di W T funzioni spiking linearmente indipendenti tramite la codifica temporale: per ogni coppia (sinapsi j, tempo t), si costruisce una funzione indicatore che vale 1 su un insieme di 2^{WT} insiemi distinti di pattern di input. L'upper bound segue dalla counting argument classico applicato alla discretizzazione degli stati della rete a T passi, con fattore log per il numero di pattern spike distinti. □

Questo risultato implica che, dal punto di vista della capacità statistica, le reti spiking non sono inferiori alle analogiche — anzi, per sequenze temporali l'efficienza per bit di informazione trasmessa è superiore nelle reti spiking, poiché ogni spike porta O(log T) bit invece di un valore reale continuo. Questa proprietà è alla base dell'efficienza energetica del calcolo neuromorfico: un chip che trasmette spike binari invece di valori floating-point riduce il consumo energetico di comunicazione di 1-3 ordini di grandezza.

14.6 Reti a Picchi di Seconda Generazione: Architetture Spike-Transformer

Le reti spiking classiche (LIF, AdEx, SRM — §13.1–13.3) presentano una limitazione nella modellazione di dipendenze temporali a lungo raggio: il kernel esponenziale della sinapsi decade con costante di tempo $\tau_{syn} \approx$ 5–20 ms, rendendo difficile la memorizzazione di contesti superiori ai 100 ms. Le Spike-Transformer Networks (STN), sviluppate da diversi gruppi nel 2023-2024, integrano il meccanismo di attenzione del transformer con la codifica spike per ottenere dipendenze temporali a lungo raggio mantenendo l'efficienza energetica del calcolo event-driven.

L'attenzione spike: SpikFormer

SpikFormer (Zhou et al., 2023) sostituisce il softmax dell'attenzione scalata con una funzione di attenzione spike-based che preserva la sparsità: date query Q, key K e value V prodotti da layer LIF, la spike attention è:

$$\text{SpikeAttn}(Q, K, V) = \text{BN}(\text{LIF}(Q K^T / \sqrt{d_k})) V$$

dove BN è la batch normalization e LIF applica la funzione di firing spike al prodotto QK^T. Poiché Q, K, V sono binari (spike), il prodotto QK^T è calcolabile tramite accumulate-and-fire invece del costoso prodotto matriciale floating-point, riducendo il costo computazionale di un ordine di grandezza rispetto al transformer standard. La sparsità tipica degli spike (~10% di attività) riduce ulteriormente il costo a ~1/100 dei FLOP del transformer denso.

Efficienza energetica: analisi formale

Per una rete spike con densità di attività r_s (frazione di neuroni che spikano per timestep), il costo energetico per operazione di attenzione è:

$$E_{spike} = E_{AC} \cdot N_{syn} \cdot r_Q \cdot r_K \cdot d_k$$

dove $E_{AC} \approx 0.1$ pJ è l'energia per operazione accumulate-and-compare su hardware neuromorfico (contro i ~4 pJ per MAC floating-point), r_Q e r_K sono le densità di spike per Q e K. Per $r_Q = r_K = 0.1$ (tipica), il risparmio energetico rispetto all'attenzione standard è circa $4 \times 100 = 400\times$ per layer. Questo rende le Spike-Transformer applicabili su hardware embedded a batteria.

Proposizione 14.2 (Complessità energetica delle Spike-Transformer).

Una Spike-Transformer Network con L layer, N_s token spike e densità di attività $r \in (0,1)$ ha costo energetico per forward pass $O(E_{AC} \cdot r^2 \cdot N_s^2 \cdot d \cdot L)$, contro $O(E_{MAC} \cdot N_s^2 \cdot d \cdot L)$ del transformer standard. Il rapporto di risparmio è $E_{AC} \cdot r^2 / E_{MAC} \approx 0.1\, r^2 / 4 \approx 0.025\, r^2 \rightarrow 0.0025$ per $r = 0.1$.

Dimostrazione. Il costo dominante è il prodotto QK^T: con spike binari $q, k \in \{0,1\}^N$, l'operazione $(q \otimes k)_i = q_i \cdot k_i$ è non nulla solo quando entrambe le componenti spikano simultaneamente, con probabilità r^2 per attività statisticamente indipendente. Il numero atteso di operazioni non nulle per prodotto è $N^2 \cdot r^2$. Ciascuna richiede E_AC invece di E_MAC. Il costo energetico totale segue moltiplicando per i layer L e la dimensione d. □

Capitolo 15 — Raccordo con il Formalismo di Lie per i Riflessi Motori

La connessione tra il calcolo neuromorfico e il formalismo dei gruppi di Lie per il controllo motorio è uno dei contributi concettuali più originali di questa monografia. Il sistema nervoso biologico implementa il controllo motorio su molteplici scale temporali: i riflessi spinali (ordine dei ms), il controllo del cerebellum (decine di ms) e la pianificazione corticale (centinaia di ms – secondi). Il formalismo di Lie offre un linguaggio unificato per descrivere tutte e tre queste scale all'interno di un'unica struttura geometrica.

15.1 Il Riflesso come Feedback Geometrico su Gruppi di Lie

Un riflesso motorio è un arco neuronale chiuso di latenza estremamente bassa (10–50 ms) che produce una risposta motoria stereotipata a uno stimolo sensoriale, senza coinvolgere il processamento corticale cosciente. Nel linguaggio del controllo geometrico, il riflesso può essere modellato come un feedback proporzionale sull'errore nello spazio delle configurazioni. Dato q ∈ Q = SE(3) × (S¹)^n come configurazione corrente dell'arto e q^{ref} il riferimento di postura, l'errore geometrico è:

$$e(t) \;=\; Log\_Q\left(q^{ref\top} \cdot q(t)\right) \in T\_e\, Q \cong \mathbb{R}\text{^}\{6 \,+\, n\} \quad (15.1)$$

dove Log_Q è il logaritmo di gruppo nella varietà Q. La legge di controllo riflessiva τ_{reflex} = −K_p e(t) − K_d ė(t), con K_p, K_d ∈ ℝ^{n×(6+n)} le matrici di guadagno, è esattamente il controllore PD intrinseco su gruppi di Lie introdotto nel contesto del WBC. Questo stabilisce che la struttura matematica del riflesso biologico e quella del controllore robotico moderno sono isomorfe.

15.2 Codifica di Popolazione su se(3) e il Teorema di Decodifica

La connessione tra il calcolo neuromorfico e il controllo geometrico emerge quando si considera la codifica neuronale dell'errore e(t) tramite popolazioni di neuroni. L'ipotesi, supportata da evidenze sperimentali

nella fisiologia del cerebellum, è che le popolazioni di neuroni Purkinje codifichino gli elementi dell'algebra di Lie se(3) tramite basi di funzioni (population coding): l'elemento ξ ∈ se(3) è rappresentato dall'attività di una popolazione di N neuroni con coefficienti di preferenza {ξ_i}_{i=1}^N come:

$$r_i = g \cdot exp\left(-\frac{\|\xi - \xi_i\|^2}{2\sigma^2}\right) + \eta_i \quad (15.2)$$

dove r_i è la frequenza di firing del neurone i, g è il guadagno, σ determina l'ampiezza del campo di tuning, ξ_i è il twist di preferenza e η_i il rumore. La decodifica (estrazione di ξ dalla popolazione) avviene tramite il decoder lineare $\hat{\xi} = \Sigma_i d_i r_i$, con coefficienti d_i determinati dalla minimizzazione dell'errore medio quadratico:

$$\{d_i\}^* = \underset{\{d_i\}}{argmin}\, E_{\xi,\eta}\left[\|\xi - \sum_i d_i r_i\|^2\right] \quad (15.3)$$

Con N ≥ (dim se(3)/σ)^{dim se(3)} neuroni uniformemente distribuiti in se(3), l'errore di decodifica è O(1/√N), rendendo la rappresentazione robusta al rumore neuronale. Questo risultato, unione del teorema di campionamento di Popoviciu e della stima dell'errore di regressione su spazi riemanniani, stabilisce il nucleo teorico della connessione tra neuromorfico e Lie.

15.3 Apprendimento On-chip e Plasticità Hebbiana su Loihi 2

Intel Loihi 2 supporta l'apprendimento on-chip tramite regole di plasticità sinaptica programmabili. La regola generale ha la forma:

$$\Delta w_{ji} = F(x_{pre,j},\ x_{post,i},\ w_{ji}) \quad (15.4)$$

dove x_{pre,j} e x_{post,i} sono vettori di tracciati locali che integrano la storia degli spike con kernel esponenziali. La programmabilità di F consente di implementare non solo STDP ma anche R-STDP (reward-modulated STDP, in cui il segnale di ricompensa modula la plasticità) e varianti dell'algoritmo e-prop (Eligibility propagation, una variante della backpropagation attraverso il tempo adattata alle reti spiking).

La convergenza dell'apprendimento on-chip è garantita, in opportune ipotesi di stazionarietà, dal teorema di convergenza delle regole di Lyapunov stocastico: se F ammette una funzione di Lyapunov strettamente decrescente V(w) lungo le traiettorie degli spike, allora i pesi convergono quasi certamente a un punto fisso w*. In pratica, le regole STDP prive di normalizzazione sono instabili (i pesi divergono per la struttura competitiva), ma regole con normalizzazione dei pesi $\Sigma_j w_{ji} = \text{const}$ o con un termine di decay $\Delta w_{ji} \to \Delta w_{ji} - \gamma\, w_{ji}$ sono convergenti con costante di tempo determinata da γ.

Questa connessione stabilisce un ponte concettuale tra il calcolo neuromorfico — implementato su hardware event-driven come Loihi 2 — e la teoria del controllo geometrico su gruppi di Lie. Il risultato pratico è che un robot con un controller neuromorfico può eseguire riflessi motori a latenza di $\approx 1\ \mu s$, tre ordini di grandezza al di sotto di un controller convenzionale su CPU/GPU, mantenendo la stessa struttura matematica del feedback geometrico su SE(3).

15.4 Il modello di Marr-Albus-Ito: apprendimento cerebellare e controllo di feedforward

Il cerebellum è una struttura neurale distinta dalla corteccia con una circuiteria altamente regolare: le cellule di Purkinje (PC) ricevono input da due fonti distinte — le fibre parallele (PF, provenienti dalle cellule dei granuli) e le fibre rampicanti (CF, provenienti dal nucleo olivare inferiore). Il modello di Marr (1969), Albus (1971) e Ito (1984) propone che il cervelletto implementi un controllore di feedforward: le cellule dei granuli codificano il contesto dello stato corrente in un'espansione ad alta dimensionalità, le cellule di Purkinje apprendono una mappa lineare da questa espansione alle correzioni motorie, e le fibre rampicanti forniscono il segnale di errore per l'apprendimento.

In notazione matematica, sia $x \in \mathbb{R}^n$ lo stato del sistema motorio (posizione e velocità articolari). Le cellule dei granuli implementano un'espansione con funzioni base $\varphi_k(x) \in \mathbb{R}$ per $k = 1, \ldots, K$ con $K \gg n$ (nel cervelletto umano, $K \approx 50$ miliardi di cellule dei granuli vs $n \approx O(10^2)$). L'output della cellula di Purkinje j è:

$$y_j(x) = \Sigma_k w_{jk} \varphi_k(x)$$

dove w_{jk} sono i pesi sinaptici PF-PC modificabili. Il segnale di errore e_j proveniente dalle fibre rampicanti (segnale di errore del movimento) aggiorna i pesi tramite la Long-Term Depression (LTD) sinaptica:

$$\Delta w_{jk} = -\eta \, e_j(t) \, \varphi_k(x(t))$$

Questa è la regola di apprendimento perceptron trasportata nello spazio cerebellar, e converge alla soluzione dei minimi quadrati $\Sigma_k w_{jk}^* \varphi_k(x) \approx f_j^*(x)$ dove f_j^* è la funzione target (la correzione di feedforward ottimale). La convergenza richiede che le funzioni base $\{\varphi_k\}$ siano sufficientemente ricche da approssimare f_j^* nello spazio funzionale — esattamente il requisito del teorema di approssimazione universale.

La connessione con il formalismo di Lie stabilita nella sezione 15.2 è che le funzioni base $\varphi_k(x)$ della codifica di popolazione su se(3) sono esattamente del tipo gaussiano su uno spazio metrico riemanniano. Il teorema di decodifica (equazione 15.3) stabilisce che la decodifica lineare $\hat{\xi} = \Sigma_k d_k r_k$ è ottimale nel senso dei minimi quadrati, e il decoder d_k è apprezzato dalla plasticità PF-PC come descritto sopra. Quindi il modello di Marr-Albus-Ito è esattamente il meccanismo biologico che implementa il decoder lineare su codifica di popolazione, unificando neurobiologia e teoria del controllo geometrico.

15.5 Controllo Neuromorfico Distribuito: Architettura Multi-Chip per Robot Umanoidi

Un robot umanoide con 30+ gradi di libertà e sensori distribuiti (forza/coppia ai polsi, pressione alle suole, vision) genera un volume di dati eventi-driven che supera la capacità di un singolo chip Loihi 2. Le architetture multi-chip — interconnesse tramite network neuromorfico on-chip — sono necessarie per distribuire il calcolo in modo che rifletta la struttura anatomica del sistema nervoso biologico: processamento spinale locale (riflessi), processamento cerebellare regionale (coordinazione), e processamento corticale globale (pianificazione).

Topologia mesh per l'elaborazione distribuita

L'interconnessione di K chip Loihi 2 in una topologia mesh 2D (√K × √K) produce un sistema con K × 10^6 neuroni totali e latenza di comunicazione inter-chip di 1–5 μs tramite il protocollo neuromorfico NxNet. La topologia ottimale per un umanoide con 30 DOF è una topologia gerarchica: cluster locali di 2–4 chip per arto (processamento dei riflessi locali), connessi a un hub centrale di 4–8 chip per la coordinazione globale.

Il routing degli spike nel mesh è gestito da tabelle di instradamento statiche pre-caricate durante la configurazione. Per minimizzare il traffico di rete — che è la bottleneck energetica del sistema multi-chip — le connessioni sinaptiche tra neuroni su chip diversi devono essere ridotte al minimo tramite metodi di partizionamento del grafo neuronale. Il problema di partizionamento ottimale è NP-hard in generale (riduzione da GRAPH-BISECTION), ma approssimato in O(n log n) tramite spectral partitioning della matrice laplaciana del grafo neuronale.

Latenza end-to-end e garanzie di tempo reale

Per un'architettura neuromorfica che implementa il controllo riflessivo su SE(3) (§15.1), la latenza end-to-end è la somma delle latenze di trasduzione, propagazione e attuazione:

$$T_{total} = T_{sensor} + T_{intra} + T_{inter} + T_{actuator}$$

dove $T_{sensor} \approx 0.1$ μs (sensore di forza a 10 MHz), $T_{intra} \approx 1–2$ μs (latenza intra-chip Loihi 2), $T_{inter} \approx 5–10$ μs (latenza inter-chip), $T_{actuator} \approx 100$ μs (latenza attuatore brushless). La latenza totale $T_{total} \approx 100–120$ μs è 10–100× inferiore rispetto a un controller basato su CPU/GPU (latenza tipica 1–10 ms), fondamentale per riflessi di protezione a impatto ad alta velocità.

Capitolo 15.6 — Vision-Language-Action Models su Hardware Neuromorfico

Raccordo con la monografia: Cap. 13–15 (calcolo neuromorfico, Loihi 2), Cap. 5–7 (VLA, LoRA, quantizzazione).

10.1 Il Problema: Latenza e Consumo Energetico dei VLA

I modelli VLA da 7B parametri hanno latenza di 100-300 ms per inferenza su GPU (Cap. 5.6, sez. Quantizzazione), incompatibile con il controllo reattivo a 50 Hz. La quantizzazione a 4 bit (GPTQ) riduce la latenza a 50-100 ms, ancora troppo lenta per i riflessi di sicurezza (< 5 ms). Il calcolo neuromorfico su Loihi 2 (Cap. 13) promette latenze nell'ordine del microsecondo, ma richiede la conversione del modello VLA in una rete spiking -- una sfida non banale.

10.2 Conversione ANN -> SNN: il Metodo Rate-Coding

La conversione di una rete neurale artificiale (ANN) in una spiking neural network (SNN) tramite rate coding (Cap. 14.1) sostituisce ogni neurone ReLU con un neurone LIF la cui frequenza di firing e proporzionale all'attivazione ReLU. Per un neurone con attivazione y = ReLU(x), la conversione produce:

tau_m * du/dt = -(u - u_rest) + R_m * I_syn(t)

r_fire = y / (tau_m * V_thresh) (frequenza di firing in Hz)

La conversione mantiene la semantica del modello originale nel limite di integrazione lungo: la media temporale dell'output spike della rete SNN converge all'output della rete ANN dopo T = 1/(r_min) ms, dove r_min e la frequenza minima di firing. Per VLA con output categorico (token di azione), T ~ 10-50 ms -- sufficiente per il controllo a 20 Hz.

10.3 Spike Transformer per VLA Neuromorfico

Il meccanismo di attenzione del transformer (Cap. 5.3) non e direttamente convertibile in SNN tramite rate coding perche il softmax richiede divisione per la somma degli esponenziali -- un'operazione non implementabile con spike binari. SpikFormer (Cap. 14.6) risolve questo problema sostituendo il softmax con spike attention:

SpikeAttn(Q, K, V) = BN(LIF(Q K^T / sqrt(d_k))) V

dove Q, K, V sono binari (spike) e il prodotto QK^T e calcolato tramite operazioni di accumulate-and-fire. Per VLA, i token visivi e linguistici vengono codificati in sequenze spike prima di essere processati dal trunk SNN.

10.4 Architettura VLA Neuromorfica Ibrida: S1 Spike + S2 GPU

L'approccio piu pratico per il deployment di VLA su hardware neuromorfico e l'architettura ibrida S1/S2 (Cap. 6.5, Helix): il livello S2 (pianificatore linguistico, 70B+ parametri) opera su GPU con latenza di 200-500 ms, mentre il livello S1 (esecutore motorio) e implementato come SNN su Loihi 2 con latenza di 1-5 ms.

Il modello SNN per S1 e un modello distillato da OpenVLA 7B: tramite knowledge distillation (Cap. 5.6), si addestra una SNN da 50M parametri equivalenti a imitare il comportamento del modello VLA completo sulle primitive motorie di base (grasp, place, move). La distillazione spike-NN ha la seguente loss:

> L_KD_spike = alpha * L_task(pi_S, D_demo) + (1-alpha) * sum_l || h_l^S - W_l * h_l^T ||^2

dove h_l^S sono le attivazioni medie (rate-coded) dello strato l della SNN e h_l^T quelle del transformer teacher. I risultati preliminari su task di manipolazione standard indicano che la SNN distillata raggiunge il 78-82% del successo del modello teacher con latenza 20-50x inferiore.

10.5 Consumo Energetico: Analisi Formale

Il consumo energetico per inferenza del VLA neuromorfico ibrido e:

> E_total = E_S2_GPU + E_S1_Loihi
>
> E_S2_GPU ~ 70B * 4bit * 2 FLOP/weight * (1 inference / 300ms) * P_GPU ~ 30 W (attivo al 10%)
>
> E_S1_Loihi ~ N_neurons * r_spike * E_AC * (1000 Hz) ~ 0.5 W

dove N_neurons = 10^6 (Loihi 2 ha 1M neuroni), r_spike = 0.1 (densita di attivita), E_AC = 0.1 pJ, e il controller S1 opera a 1 kHz. Il consumo totale e dominato da S2 durante la pianificazione (30 W attivo) e da S1 durante l'esecuzione (0.5 W continuo). Rispetto a un sistema puramente GPU (100-200 W), il risparmio energetico e di 5-10x per task con brevi periodi di pianificazione e lunghi periodi di esecuzione.

Conclusioni — Convergenza dei Paradigmi e Prospettive Future

La rassegna condotta in questa monografia rivela una convergenza concettuale profonda tra paradigmi che, a prima vista, appaiono radicalmente distinti. Il controllo whole-body degli umanoidi, l'apprendimento di policy tramite modelli visio-linguistici, la pianificazione tramite world models fisici, l'imitazione a basso costo e il calcolo neuromorfico trovano un substrato matematico comune nei gruppi di Lie, nel trasporto ottimale e nella geometria delle misure. Questa unità non è puramente formale: si manifesta in connessioni operative — il flow matching come prior su traiettorie latenti nei world models, la distanza di Wasserstein come metrica sia nei JEPA sia nella simulazione differenziabile, il logaritmo di gruppo come misura dell'errore sia nel WBC sia nella codifica neuronale su se(3) — che suggeriscono la presenza di strutture profonde ancora parzialmente inesplorate.

La traiettoria più rilevante per il futuro è l'integrazione di questi paradigmi in sistemi gerarchici in cui ogni livello del controllo è implementato nel substrato computazionale più appropriato: il controllo riflessivo a latenza ultra bassa su chip neuromorfi come Loihi 2, il controllo di postura e pianificazione locale su chip robotici dedicati, e la pianificazione semantica e linguistica su acceleratori GPU. Questa stratificazione riflette la struttura del sistema nervoso biologico — dove riflessi spinali, controllo cerebellare e pianificazione corticale operano a latenze crescenti su substrati dedicati — e suggerisce che la convergenza tra neurobiologia e robotica non è meramente analogica ma strutturale.

Sul piano matematico, i problemi aperti più rilevanti riguardano tre aree distinte. La prima è la teoria dell'apprendimento su varietà non compatte come SE(3)^n: le stime di generalizzazione disponibili per VC-dimension e Rademacher complexity si applicano a spazi vettoriali di dimensione finita e richiedono estensioni non banali per varietà riemanniane. La seconda è la simulazione differenziabile per sistemi con contatti discontinui: nonostante i progressi recenti con Genesis e MuJoCo MJX, la derivata delle mappe di contatto rimane non definita nelle superfici di

transizione, e non esiste ancora una teoria unificata che garantisca la convergenza degli algoritmi di ottimizzazione in presenza di queste discontinuità. La terza è la teoria dell'apprendimento per reti spiking con struttura di Lie: la connessione stabilita tra population coding su se(3) e controllo geometrico apre un programma di ricerca in cui le proprietà della rappresentazione neuronale — efficienza, capacità, robustezza al rumore — si rispecchiano nelle proprietà del controllore geometrico corrispondente.

La prospettiva concreta è quella di un robot umanoide domestico che integra tutti i moduli discussi: un controllore WBC-MPC in tempo reale a 1 kHz per la stabilità posturale, un modello VLA da 7B parametri per la comprensione delle istruzioni e la pianificazione manipolativa, un world model fisico per la previsione delle conseguenze delle azioni, una policy di imitazione addestrata con dimostrazioni a basso costo per i compiti specifici del dominio, e un controllore neuromorfico per i riflessi di protezione e sicurezza. La distanza tra questo scenario e le piattaforme attuali — Figure 02, Optimus, Atlas, Digit — è misurabile non in decenni ma in anni, a patto che i progressi nel hardware neuromorfico, nella simulazione differenziabile e nell'apprendimento cross-embodiment mantengano il ritmo degli ultimi tre anni.

Parte VI

Sicurezza Certificata, Control Barrier Functions e Robustezza Avversariale

Parte VI — Sicurezza Certificata per Sistemi Robotici Intelligenti

La sicurezza operativa dei robot umanoidi in ambienti condivisi con esseri umani pone requisiti qualitativamente diversi da quelli della performance: non basta che il sistema si comporti bene in media, ma occorre garantire matematicamente che certe condizioni (no-collisioni, no-falls, no-danneggiamento di oggetti) siano rispettate con certezza — o almeno con probabilità alta e verificabile. Questa parte sviluppa tre approcci complementari alla sicurezza certificata: le **Control Barrier Functions** (CBF) per la safety in tempo reale, la **Hamilton-Jacobi reachability** per la sicurezza su orizzonti lunghi, e l'analisi della **robustezza avversariale** per i modelli VLA esposti a perturbazioni negli input sensoriali.

Capitolo 16 — Control Barrier Functions e Sintesi di Controllori Safe

16.1 Invarianza Positiva e il Teorema di Nagumo

Il concetto fondamentale per la sicurezza in tempo reale è l'**invarianza positiva**: un insieme $C \subset R^n$ è positivamente invariante per il sistema $\dot{x} = f(x,u)$ se ogni traiettoria che inizia in C rimane in C per ogni tempo futuro. Per i robot umanoidi, C può rappresentare la regione dello spazio di stato in cui il robot non si trova in configurazione pericolosa (distanza dai muri, angoli articolari nei limiti, ZMP dentro il poligono di supporto).

Sia $h: R^n \to R$ una funzione smooth. Il **safe set** è $C = \{x: h(x) \geq 0\}$ con interior $Int(C) = \{x: h(x) > 0\}$ e boundary $\partial C = \{x: h(x) = 0\}$.

Teorema 16.1 (Nagumo 1942). *Il safe set $C = \{x: h(x) \geq 0\}$ è positivamente invariante per il sistema $\dot{x} = f(x)$ se e solo se:*

$$\nabla h(x) \cdot f(x) \geq 0 \quad \forall x \in \partial C$$

Dimostrazione. Per la sufficienza: la condizione implica che sulla boundary ∂C, la derivata temporale $\dot{h} = \nabla h \cdot f \geq 0$. Quindi h non può diventare negativa attraverso ∂C, e l'insieme è invariante.

Per la necessità: se la condizione è violata in $x^* \in \partial C$, allora $\dot{h}(x^*) = \nabla h(x^*) \cdot f(x^*) < 0$, il che significa che la traiettoria $x(t)$ con $x(0) = x^*$ soddisfa $h(x(\epsilon)) < 0$ per $\epsilon > 0$ piccolo, uscendo da C. ▫

16.2 Control Barrier Functions: Definizione e Proprietà

La condizione di Nagumo si applica a sistemi autonomi. Per sistemi controllati $\dot{x} = f(x) + g(x)u$, la condizione deve essere modificata per ammettere l'azione del controllo.

Definizione 16.1 (Control Barrier Function). *Una funzione $h: R^n \to R$ è una **Control Barrier Function** (CBF) per il sistema $\dot{x} = f(x) + g(x)u$ rispetto al safe set C se esiste una funzione di classe K_∞ α tale che:*

$$sup_{u\in U}\left[L_f h(x) + L_g h(x)u\right] \geq -\alpha(h(x)) \quad \forall x \in R^n$$

dove $L_f h = \nabla h \cdot f$ *e* $L_g h = \nabla h \cdot g$ *sono le derivate di Lie di* h *lungo* f *e* g*.*

Teorema 16.2 (Sicurezza con CBF, Ames et al. 2017). *Se* h *è una CBF per il sistema controllato e il controllo* $u^*(x)$ *soddisfa:*

$$L_f h(x) + L_g h(x) u^*(x) \geq -\alpha(h(x)) \quad \forall x$$

allora il safe set C *è forward invariante: se* $x(0) \in C$ *allora* $x(t) \in C$ *per ogni* $t \geq 0$*.*

Dimostrazione. Lungo ogni traiettoria del sistema, $\dot{h}(x(t)) = L_f h + L_g h u^* \geq -\alpha(h)$. Questa è una disequazione differenziale ordinaria $\dot{h} \geq -\alpha(h)$ con $h(0) \geq 0$. Per il teorema di confronto di Gronwall, la soluzione di $\dot{v} = -\alpha(v)$ con $v(0) = h(0) \geq 0$ soddisfa $v(t) \geq 0$ per ogni $t \geq 0$ (per classe K_∞). Quindi $h(x(t)) \geq v(t) \geq 0$. ▫

16.3 QP Unificato CLF-CBF per Robot Umanoidi

Il WBC del Capitolo 2 calcola le coppie τ che realizzano un insieme di task cinematici e dinamici. La safety con CBF aggiunge i vincoli di barrier come ulteriori constraint del QP. Il **QP unificato CLF-CBF** è:

$$\min_{\ddot{\delta\eta},\tau,\lambda,\delta} \quad \underbrace{\left\| J_k \ddot{\delta\eta} - \dot{x}_k \right\|^2}_{\hookrightarrow task\ error} + p\delta^2$$

soggetto a:

$$M(\eta)\ddot{\delta\eta} + h\left(\eta, \dot{\eta}\right) = S\tau + J_c^\top \lambda \quad (dinamiche)$$

$$L_f^2 h_i + L_g h_i \left(M^{-1}\left(S\tau + J_c^\top \lambda\right)\right) \geq -\alpha_2\left(L_f h_i\right) - \alpha_1\left(h_i\right) \quad (CBF,\ per\ ogni\ i)$$

$$V_{task} \leq -\delta \quad (CLF\ con\ slack\ \delta \geq 0)$$

dove h_i è la i-esima barrier function (distanza dai muri, angoli articolari, ecc.), la condizione CBF è estesa al secondo ordine per sistemi di grado relativo 2 (tipico per robot umanoidi dove la safety dipende dalla posizione, non dalla velocità), e δ è una variabile di slack che permette al

controllore di sacrificare il raggiungimento del task per mantenere la safety.

Proposizione 16.1 (Consistenza CLF-CBF). *Il QP unificato CLF-CBF ha soluzione fattibile se e solo se il set di vincoli CBF è compatibile con i vincoli dinamici del sistema. Una condizione sufficiente di compatibilità è che il set* $\{x: h_i(x) \geq 0 \ \forall i\}$ *sia in generale posizione rispetto ai vincoli cinematici (nessuna configurazione CBF-attiva è singolare per* J_c*).*

CBF per la Safety del ZMP e dell'End-Effector

Per un robot umanoide in locomozione, le CBF più rilevanti sono:

1. **CBF del ZMP**: $h_{ZMP}(x) = d\left(ZMP, \partial P_{support}\right)$ (distanza dal bordo del poligono di supporto). La condizione CBF $\dot{h}_{ZMP} \geq -\alpha\left(h_{ZMP}\right)$ impone che il ZMP non si avvicini al bordo troppo rapidamente.
2. **CBF di non-collisione**: $h_{coll,i}(x) = \|p_{link,i} - p_{obs}\| - d_{safe}$ (distanza dal link i all'ostacolo, meno una distanza di sicurezza). Per ostacoli dinamici, la CBF include la velocità relativa: $h_{coll,i}(x) = \|p_{rel}\| - d_{safe} + \frac{p_{rel}^{\top} v_{rel}}{\|p_{rel}\|} T_s$ con T_s orizzonte di sicurezza.
3. **CBF degli angoli articolari**: $h_{joint,k}(x) = \left(q_k - q_{k,min}\right)\left(q_{k,max} - q_k\right)$ che è positiva nell'interno dell'intervallo e zero ai limiti.

16.4 CBF di Ordine Elevato (HOCBF) per Sistemi con Grado Relativo > 2

Le CBF del Capitolo 16 assumono grado relativo uno rispetto al controllo u: L_g h(x) ≠ 0, ovvero l'ingresso appare direttamente nella derivata prima di h. Per molte constraint robotiche (distanze tra link che dipendono dalla posizione, non dalla velocità o accelerazione), il grado relativo è 2 o superiore, rendendo inapplicabile direttamente la definizione standard. Le High-Order Control Barrier Functions (HOCBF, Xiao & Belta, 2021) estendono la teoria alle constraint di grado relativo arbitrario.

Definizione e costruzione della HOCBF

Sia h(x) una funzione di sicurezza con grado relativo m rispetto all'ingresso u (ovvero L_g h = L_g L_f h = ... = L_g L_f^{m-2} h = 0 e L_g L_f^{m-1} h ≠ 0). Si definisce la sequenza di funzioni ausiliarie:

$$\psi_0(x) = h(x)$$

$$\psi_1(x) = \dot{\psi}_0(x) + \alpha_1(\psi_0(x)) = L_f\, h(x) + \alpha_1(h(x))$$

$$\psi_2(x) = \dot{\psi}_1(x) + \alpha_2(\psi_1(x)) = L_f^2 h(x) + (\partial\alpha_1/\partial h)\, L_f\, h(x) + \alpha_2(\psi_1(x))$$

...

$$\psi_{m-1}(x) = \dot{\psi}_{m-2}(x) + \alpha_{m-1}(\psi_{m-2}(x))$$

dove α_m sono funzioni di classe K_∞. La condizione HOCBF di ordine m richiede:

$$L_f^m\, h(x) + L_g\, L_f^{m-1}\, h(x)\, u + O(\psi_0,\ldots,\psi_{m-1}) \geq -\alpha_m(\psi_{m-1}(x))$$

Teorema 16.3 (Invarianza del safe set con HOCBF, Xiao-Belta 2021).

Se la funzione h(x) con grado relativo m soddisfa la condizione HOCBF e le funzioni ausiliarie $\{\psi_1,\ldots,\psi_{m-1}\}$ sono inizialmente positive, allora l'insieme $C = \{x: h(x) \geq 0\}$ è forward invariante.

Dimostrazione. La dimostrazione procede per induzione sulle funzioni ausiliarie: la condizione $\psi_{m-1}(x) \geq 0$ è mantenuta per costruzione dalla condition HOCBF. Dato che $\psi_{m-1} \geq 0$ e α_{m-1} è di classe K_∞, si ha che $\psi_{m-2}(x) \geq 0$ è mantenuta tramite la struttura a cascata. L'invarianza risale fino a $\psi_0 = h$, completando la dimostrazione. □

Applicazione alla distanza sicura tra link del robot

Per la constraint di non-collisione tra il link i del robot e un ostacolo, la funzione gap $\varphi_{coll}(q) = \|p_{link,i}(q) - p_{obs}\| - d_{safe}$ dipende dalla posizione del link (che dipende da q), ma non direttamente dalle coppie u. Il grado relativo è m = 2 (la coppia appare nella derivata seconda di φ_{coll} tramite l'accelerazione del link). La HOCBF di secondo ordine produce la constraint:

$$\dot{W}_{coll}(q,\dot{q})\, u \geq -\ddot{\varphi}_{coll,free} - \alpha_2(\dot{\varphi}_{coll} + \alpha_1(\varphi_{coll}))$$

dove Ẇcoll è lo Jacobiano analitico della velocità del link e φ̈_coll_free è il termine libero dell'accelerazione. Questo vincolo viene aggiunto al QP gerarchico del WBC (§16.3) come constraint di disuguaglianza lineare in u, senza aumentare l'ordine del problema.

Capitolo 16.5 — Composizione di Control Barrier Functions e Analisi della Compatibilità

Raccordo con la monografia: Cap. 16 (CBF, QP unificato), Cap. 16.4 (HOCBF).

9.1 Il Problema della Composizione di Barriere

Il QP unificato CLF-CBF del Cap. 16.3 include le CBF come vincoli di disuguaglianza. In pratica, un robot umanoide in operazione deve soddisfare simultaneamente molteplici constraint di safety: non-collisione con gli oggetti della scena (CBF_coll), ZMP dentro il poligono di supporto (CBF_ZMP), limiti agli angoli articolari (CBF_joint), limiti alle coppie (CBF_torque). La composizione di piu CBF introduce il rischio di incompatibilita: il QP potrebbe non avere soluzioni fattibili quando piu barriere sono attive simultaneamente.

9.2 Condizioni di Compatibilita per Coppie di CBF

Definizione S9.1 (Compatibilita di CBF).

Due CBF h_1 e h_2 sono compatibili se l'insieme safe composito C_12 = {x: h_1(x) >= 0, h_2(x) >= 0} e forward invariante sotto la legge di controllo u* = argmin_{u in U} ||u - u_nom||^2 s.t. CBF_1, CBF_2.

Teorema S9.1 (Condizione sufficiente di compatibilita per CBF lisce).

Siano h_1, h_2 due CBF con alpha_1, alpha_2 di classe K_infty. Un'insieme sufficiente di condizioni per la compatibilita e che sul bordo condiviso Delta_12 = {x: h_1(x) = 0, h_2(x) = 0}:

$$\max_{u \in U} [L_f h_1(x) + L_g h_1(x)\, u] \geq -\alpha_1(h_1(x))$$

$$\max_{u \in U} [L_f h_2(x) + L_g h_2(x)\, u] \geq -\alpha_2(h_2(x))$$

$$\text{AND} \quad \max_{u \in U} \min(L_g h_1(x)\, u, L_g h_2(x)\, u) > 0$$

La terza condizione richiede che esista un ingresso u che incrementa simultaneamente h_1 e h_2 -- ovvero che le direzioni di incremento delle due barriere non siano diametralmente opposte.

Dimostrazione. Sulla frontiera Delta_12, le condizioni CBF richiedono L_g h_i(x) u >= -L_f h_i(x) per i=1,2. Per la compatibilita, deve esistere u in U che soddisfa entrambi i vincoli. La terza condizione garantisce che le due half-plane di vincolo su u si intersecano non vuotamente. La forward invarianza di C_12 segue allora dall'applicazione individuale del Teorema 16.2 per ciascuna barriera con l'ingresso compatibile. []

9.3 Composizione in Serie: Barriere Concatenate

Un approccio alternativo alla composizione e il concatenamento di barriere: si costruisce una barriera composita h_comp dalla composizione booleana delle singole barriere. Per l'intersezione (AND logico):

h_AND(x) = min(h_1(x), h_2(x)) (soft: h_AND(x) = -1/mu * log(e^{-mu h_1} + e^{-mu h_2}))

Per l'unione (OR logico):

h_OR(x) = max(h_1(x), h_2(x)) (soft: h_OR(x) = 1/mu * log(e^{mu h_1} + e^{mu h_2}))

La versione soft (log-sum-exp) e differenziabile ovunque, consentendo il calcolo di L_g h_comp tramite la regola della catena. La versione min esatta non e differenziabile dove h_1 = h_2, ma puo essere gestita tramite il subgradiente.

9.4 Risoluzione dei Conflitti tramite Gerarchia di Priorita

Quando due CBF sono in conflitto -- non esiste u che soddisfi entrambi i vincoli -- e necessario definire una gerarchia di priorita. La struttura naturale e la seguente: safety fisica (no-collisione, joint limits) > stabilita posturale (ZMP) > task performance (raggiungimento del goal). Il QP a priorita risolve i vincoli nell'ordine:

min_u ||u - u_nom||^2

s.t. CBF_phys(x, u) >= 0   (priorita 1: safety fisica, hard)

  CBF_ZMP(x, u) >= -delta_1 (priorita 2: stabilita, con slack delta_1 >= 0)

  CBF_task(x, u) >= -delta_2 (priorita 3: task, con slack delta_2 >= 0)

dove i vincoli di priorita inferiore vengono rilassati tramite variabili di slack quando incompatibili con quelli di priorita superiore. Questo schema e duale all'HQP del WBC (Cap. 2.2) applicato agli obiettivi di safety invece che ai task cinematici.

Capitolo 17 — Hamilton-Jacobi Reachability per Sicurezza su Orizzonte Lungo

17.1 L'Insieme Reachable e il Problema HJI

La reachability analysis risponde alla domanda: da quali stati iniziali il sistema può raggiungere una regione "unsafe" in un orizzonte temporale $[0, T]$, indipendentemente dal controllo applicato (o, dualmente, da quale insieme il sistema può essere **garantito** in sicurezza da un controllore difensivo)?

Sia $T \subset R^n$ l'insieme target (unsafe). Il **backward reachable set** $R(t, T)$ è l'insieme degli stati da cui il sistema può essere forzato dentro T entro il tempo t dal disturbo, nonostante il tentativo del controllore di evitarlo:

$$R(t,T) = \{x_0 : \exists d(\cdot),\ \forall u(\cdot),\ \exists s \in [0,t]\ t.c.\ x\left(s; x_0, u, d\right) \in T\}$$

Il backward reachable set è caratterizzato dalla **viscosity solution** $V : R^n \times [0,T] \to R$ dell'**equazione HJI** (Hamilton-Jacobi-Isaacs):

$$\frac{\partial V}{\partial t} + min\left[0,\ \min_u\ \max_d\ H\left(x, \nabla_x V, u, d\right)\right] = 0$$

$$V(x,0) = l(x) \quad (condizione\ terminale,\ l(x) < 0 \Leftrightarrow x \in T)$$

dove $H(x,p,u,d) = p^\top f(x,u,d)$ è l'Hamiltoniana del sistema con stato x, gradiente $p = \nabla_x V$, controllo u e disturbo d. Il backward reachable set è $R(t,T) = \{x : V(x,t) \leq 0\}$.

Teorema 17.1 (Mitchell et al. 2005). *La viscosity solution V dell'equazione HJI caratterizza il backward reachable set: $x \notin R(T,T)$ se e solo se $V(x,T) > 0$. Inoltre, il controllore safe ottimale è:*

$$u^*(x,t) = arg\ \min_{u \in U}\ H\left(x, \nabla_x V(x,t), u, d^*(x,t)\right)$$

dove $d^*(x,t) = argmax_d H\left(x, \nabla_x V, u^*, d\right)$ è il disturbo peggiore.

17.2 Level Set Methods per la Soluzione Numerica dell'HJI

La soluzione numerica dell'equazione HJI richiede la discretizzazione dello spazio degli stati su una griglia n-dimensionale, con complessità $O\left(N^n\right)$ per una griglia $N\times\cdots\times N$ in n dimensioni. Per $n = 6$ (pelvi del robot) con $N = 50$, questo richiede $50^6 \approx 1.5\times10^{10}$ celle — computazionalmente proibitivo per tempo reale.

Soluzioni al Problema Dimensionale

Decomposizione in sottoproblemi: Per sistemi con struttura quasi-decoupled (dinamica centroidale separata dagli angoli articolari), il problema HJI si decompone in sottoproblemi di dimensione ridotta. Ad esempio, la safety della camminata può essere analizzata nel sottospazio 3D $\left(x_{CoM}, \dot{x}_{CoM}, \theta_{trunk}\right)$ con un problema HJI 3D risolvibile su griglia $50^3 \approx 125000$ celle.

Safety Index Function: Si costruisce un safety index $\psi(x)$ che approssima la viscosity solution V con una funzione analitica dipendente dalla configurazione. La funzione più comune per umanoidi è:

$$\psi(x) = \sum_i w_i h_i(x) - V_{threshold}$$

dove h_i sono CBF individuali pesate. Il controllore safe usa ψ al posto della soluzione HJI esatta, con un bound di sub-ottimalità espresso dalla norma $\|\psi - V\|_\infty$.

Deep HJI: Mitchell et al. (2021) e Bansal et al. (2021) propongono di approssimare la viscosity solution V con una rete neurale addestrata tramite l'equazione HJI come loss:

$$L_{HJI}(\theta) = E_x\left[\left|\frac{\partial V_\theta}{\partial t} + min\left(0, \min_u \max_d H\left(x, \nabla_x V_\theta, u, d\right)\right)\right|^2\right]$$

I risultati sperimentali mostrano che reti con 5–7 layer e 512 unità approssimano V con errore $\|\hat{V} - V\|_\infty \leq 0.05$ in sistemi fino a 10

dimensioni, con tempi di forward pass di 1–5 ms — compatibili con il ciclo di controllo del WBC.

17.3 Connessione tra HJI Reachability e CBF

Proposizione 17.1 (CBF come certificato di reachability). *Se* $h(x) > 0$ *implica* $x \notin R(T, T)$ *per ogni* $T > 0$*, allora il safe set* $C = \{x : h(x) \geq 0\}$ *è* ***reach-avoid safe****: il sistema che rispetta la condizione CBF non raggiunge mai la regione unsafe, su qualsiasi orizzonte temporale.*

Dimostrazione. La forward invarianza di C (Teorema 16.2) implica che $x(t) \in C$ per ogni t se $x(0) \in C$. Per costruzione, $C \cap R(T, T) = \emptyset$ (ipotesi), quindi $x(t) \in C$ implica $x(t) \notin T$ per ogni $T > 0$. ▫

Capitolo 18 — Robustezza Avversariale dei Modelli VLA

18.1 Attacchi Avversariali agli Input Visivi dei VLA

I modelli VLA sono vulnerabili agli **attacchi avversariali**: perturbazioni δ di norma piccola nell'immagine di input I che inducono azioni pericolose $a = \pi(I + \delta)$. A differenza degli attacchi sui classificatori (dove l'obiettivo è cambiare la classe predetta), gli attacchi su VLA possono puntare a indurre azioni specifiche:

$$\delta^{*} = arg \max_{\|\delta\|_p \leq \varepsilon} L\left(\pi_\theta(I + \delta), a_{target}\right)$$

dove a_{target} è l'azione target dell'attaccante e L è una loss di matching.

Il **FGSM** (Fast Gradient Sign Method, Goodfellow et al. 2015) produce la perturbazione:

$$\delta_{FGSM} = \varepsilon \cdot sign\left(\nabla_I L\left(\pi_\theta(I), a_{target}\right)\right)$$

in un singolo step del gradiente. Il **PGD** (Projected Gradient Descent, Madry et al. 2018) itera k passi con proiezione sulla palla ℓ_p:

$$\delta^{(t+1)} = \Pi_{\|\delta\|_p \leq \varepsilon}\left[\delta^{(t)} + \alpha \cdot sign\left(\nabla_I L\left(\pi_\theta\left(I + \delta^{(t)}\right), a_{target}\right)\right)\right]$$

La robustezza avversariale di un modello VLA si misura con l'**errore avversariale**:

$$\varepsilon_{adv}(\pi, \varepsilon) = E_{(I,a^{*})\sim D}\left[sup_{\|\delta\|_p \leq \varepsilon} L\left(\pi(I + \delta), a^{*}\right)\right]$$

18.2 Lipschitz Bounding per Transformer e Certified Robustness

La **robustezza certificata** fornisce un bound on the adversarial error che è verificabile matematicamente, senza richiedere la ricerca esplicita del worst-case attack.

Definizione 18.1 (Costante di Lipschitz di un transformer). *Un transformer* $f: X \to Y$ *è L-Lipschitz nella norma* $\|\cdot\|_p$ *se* $\|f(x) - f(x')\|_q \leq L\|x - x'\|_p$ *per ogni* $x, x' \in X$.

Proposizione 18.1 (Bound di Lipschitz per layer di attenzione). *Un layer di attenzione scalata* $Attn(X) = softmax\left(QK^\top/\sqrt{d_k}\right)V$ *con* $Q = XW^Q, K = XW^K, V = XW^V$ *ha costante di Lipschitz:*

$$L_{Attn} \leq \|W^V\|_2 \cdot \left(1 + \frac{\|W^Q\|_2 \|W^K\|_2}{2\sqrt{d_k}}\right)$$

nella norma spettrale (norma dell'operatore $\ell^2 \to \ell^2$*).*

Dimostrazione. Il layer di attenzione è la composizione di operazioni: la proiezione lineare (Q, K, V), il prodotto scalare scalato $QK^\top/\sqrt{d_{k'}}$ e la moltiplicazione softmax $\times$ V. La costante di Lipschitz del prodotto scalare è $\|W^Q\|_2 \|W^K\|_2/\left(2\sqrt{d_k}\right)$ (per la disuguaglianza di sub-moltiplicatività e la Lipschitz-continuità del softmax con costante 1/2). Componendo con la proiezione V si ottiene il bound. ▫

Per un transformer con L layer, la costante di Lipschitz composita è $\prod_{l=1}^{L} L_{Attn'}^{(l)}$ che cresce esponenzialmente con la profondità. Questo spiega la vulnerabilità degli LLM profondi agli attacchi avversariali e motiva le tecniche di lipschitz regularization.

18.3 Randomized Smoothing e Certified Robustness per Policy Robotiche

Il **randomized smoothing** (Cohen et al. 2019) è la tecnica attualmente più scalabile per la certified robustness, applicabile a qualsiasi modello black-box. L'idea è di costruire una versione "smoothed" della policy π^- che agisce sulla media delle azioni prodotte da versioni rumorose dell'osservazione:

$$\pi^-(I) = E_{\eta \sim N\left(0, \sigma^2 I\right)}[\pi(I + \eta)]$$

Teorema 18.1 (Certified Robustness via Randomized Smoothing, Cohen et al. 2019). *Sia* π^{-} *la policy smoothed e sia* $a = \pi^{-}(I)$. *Se:*

$$P[\pi(I + \eta) = a] \geq p_A > 1/2$$

allora $\pi^{-}(I + \delta) = a$ *per ogni* $\|\delta\|_2 \leq R = \sigma\Phi^{-1}(p_A)$, *dove* Φ^{-1} *è l'inverso della CDF gaussiana.*

Dimostrazione. La prova usa il teorema di Neyman-Pearson: per ogni classificatore alternativo che produca una risposta diversa da a, il test basato sul likelihood ratio delle distribuzioni $N(I, \sigma^2 I)$ e $N(I + \delta, \sigma^2 I)$ ha un unique most powerful region. Il raggio $R = \sigma\Phi^{-1}(p_A)$ è esattamente il raggio entro cui il classificatore non può essere cambiato da perturbazioni ℓ^2 senza abbassare p_A sotto $1/2$. ▫

Limitazione per azioni continue. Il Teorema 18.1 è formulato per classificatori discreti. Per policy robotiche con azioni continue $a \in R^m$, la certified robustness si formula come bound sulla distanza dell'azione perturbata dall'azione nominale:

$$\|\pi^{-}(I + \delta) - \pi^{-}(I)\|_2 \leq L_{smooth}\|\delta\|_2$$

dove $L_{smooth} = 1/\sigma$ è la costante di Lipschitz della policy smoothed (per input gaussiani). Il parametro σ controlla il tradeoff tra robustezza (σ grande → maggiore raggio certificato) e accuratezza (σ grande → più rumore sulle azioni nominali).

18.4 Adversarial Training per VLA e Integrazione con CBF

L'**adversarial training** (Madry et al. 2018) rende il modello robusto agli attacchi PGD addestrandolo su esempi avversariali:

$$\min_{\theta} E_{(I,a^*)\sim D}\left[\max_{\|\delta\|_p \leq \varepsilon} L(\pi_\theta(I + \delta), a^*)\right]$$

Il problema interno (max) è risolto con k passi di PGD, il problema esterno (min) con SGD. L'adversarial training aumenta ε_{adv} al costo di un degrado

della performance nominale, quantificabile tramite il **robustness-accuracy tradeoff**:

Proposizione 18.2 (Tradeoff robustezza-accuratezza per VLA). *Sia* ε_{nom} *l'errore nominale e* $\varepsilon_{adv}(\varepsilon)$ *l'errore avversariale al raggio* ε*. Il tradeoff soddisfa:*

$$\varepsilon_{adv}(\varepsilon) + \varepsilon_{nom} \geq P_{(I,a^*)\sim D}\left[\| a^* - a^- \|_2 \geq \varepsilon/(2L)\right]$$

dove a^- *è l'azione più comune nella distribuzione D e L è la costante di Lipschitz del modello. Il bound è positivo se la distribuzione delle azioni è non banale.*

L'integrazione tra adversarial robustness e CBF produce il **safe adversarial control**: il controllore QP unificato (Sezione 16.3) include i vincoli CBF di sicurezza come hard constraints, mentre il modello VLA usa randomized smoothing per la robustezza avversariale. Le azioni prodotte dal VLA avversarialmente robusto vengono filtrate attraverso il QP CBF prima di essere inviate agli attuatori, garantendo sia la robustezza all'input noise sia la safety certificata.

18.6 Certifiable Safety tramite Sum-of-Squares e Barriere Polinomiali

Le CBF e le HOCBF garantiscono la safety solo localmente, nell'intorno dell'equilibrio o in regioni raggiungibili. Per ottenere garanzie globali di safety verificabili computazionalmente, si adottano metodi basati sulla programmazione semidefinita (SDP) e i Sum-of-Squares (SOS) polynomials, che forniscono certificati di sicurezza validi sull'intero spazio di stato.

Programmazione SOS per barrier certificates

Sia $x \in \mathbb{R}^n$ e il sistema $\dot{x} = f(x, u)$. Un barrier certificate è una funzione B(x) tale che: (1) $B(x) \leq 0$ per $x \in X_0$ (stato iniziale safe), (2) $B(x) > 0$ per $x \in X\_unsafe$ (stati non sicuri), (3) $dB/dt \leq 0$ lungo le traiettorie del sistema (B è decrescente). L'esistenza di un barrier certificate implica che le traiettorie da X_0 non raggiungono mai X_unsafe. Per sistemi polinomiali f(x,u) e constraint polinomiali, la verifica dell'esistenza di B(x) polinomiale si formula come un problema SDP tramite il Positivstellensatz di Stengle:

Un polinomio p(x) è non-negativo su un insieme algebrico X = {x: g⊚(x) ≥ 0} se esiste una combinazione SOS: p(x) = σ_0(x) + Σ⊚ σ⊚(x) g⊚(x) dove σ⊚ sono polinomi SOS (somme di quadrati). La verifica dell'esistenza di B(x) SOS con grado al più 2d si riduce a un SDP di dimensione O(n^{2d}), risolvibile in tempo polinomiale con solver come MOSEK o SDPT3.

Connessione con i Neural Lyapunov Methods

L'approccio SOS ha il limite di scalare esponenzialmente con la dimensione n del sistema — già per n = 20 (sottosistema centroidale) il SDP è computazionalmente pesante. I Neural Lyapunov Methods (Chang et al., 2020; Dawson et al., 2023) sostituiscono il barrier certificate polinomiale con una rete neurale B_θ(x) e verificano le condizioni di safety tramite satisfiability modulo theories (SMT) o tramite counterexample-guided synthesis. La rete è addestrata su un dataset di stati {x_i} con una loss che penalizza le violazioni delle condizioni di barrier:

$$L_{barrier}(\theta) = \Sigma_{i \in X_0} \max(0, B_\theta(x_i)) + \Sigma_{i \in X_{unsafe}} \max(0, -B_\theta(x_i)) + \Sigma_{i \in X} \max(0, \partial B_\theta / \partial x \, f(x_i, u_i))$$

Il terzo termine penalizza i punti dove il barrier cresce lungo le traiettorie del sistema. Dopo l'addestramento, la verifica formale è eseguita tramite branch-and-bound o Lipschitz-based bounds sul residuo della loss, fornendo una garanzia formale con errore quantificabile.

Parte VII

Task and Motion Planning e Architetture Neurosimboliche

Parte VII — Pianificazione Simbolica, Grounding e Architetture Neurosimboliche

I capitoli precedenti hanno trattato policy reattive: dato un osservazione, produce un'azione. Ma molti compiti robotici richiedono una pianificazione su sequenze di azioni che hanno struttura simbolica (prendere il bicchiere, poi portarlo al tavolo, poi lasciarlo) e che devono essere eseguite nel mondo continuo rispettando vincoli geometrici. Il **Task and Motion Planning** (TAMP) è il framework che integra questa pianificazione simbolica con la pianificazione geometrico-cinematica continua. Il recente avvento degli LLM ha poi aperto una nuova direzione: utilizzare modelli linguistici come pianificatori simbolici che generano piani in linguaggio naturale o codice, poi eseguiti da controller geometrici.

Capitolo 19 — Task and Motion Planning: PDDL, PDDLStream e Complessità

19.1 Fondamenti del Task Planning: STRIPS e PDDL

Il **Task Planning** classico lavora in un dominio discreto: un insieme finito di **predicati** $P = \{P_1, ..., P_k\}$ che descrivono lo stato del mondo, un insieme di **azioni** $A = \{a_1, ..., a_m\}$ ciascuna con precondizioni e effetti, e un obiettivo espresso come congiunzione di predicati.

Il linguaggio **PDDL** (Planning Domain Definition Language) formalizza questo framework. Un **dominio PDDL** è una tupla $\langle T, P, A \rangle$ dove T è l'insieme dei tipi, P i predicati tipati e A le azioni schema. Un'**istanza** aggiunge gli oggetti O, lo stato iniziale $s_0 \subset P \times O^*$ e il goal g.

Definizione 19.1 (Piano PDDL). *Un piano* $\pi_{PDDL} = (a_1, ..., a_n)$ *è una sequenza di azioni ground (istanziate) che, applicate allo stato iniziale* s_0*, producono uno stato finale che soddisfa il goal* g*.*

Teorema 19.1 (Complessità del PDDL classico). *Il problema della pianificazione STRIPS è: (i) PSPACE-completo in generale. (ii) NP-completo per problemi con profondità del piano polinomialmente bounded. (iii) Risolvibile in tempo polinomiale per domini con azioni non interagenti (trivially separable).*

Dimostrazione (sketch). Il lower bound PSPACE segue dalla riduzione polinomiale dal problema QBF (Quantified Boolean Formula), che è PSPACE-completo, al problema STRIPS con spazio degli stati esponenziale. La contenimento in PSPACE segue dal fatto che una sequenza di stati nel piano occupa spazio polinomiale (si verifica un piano alla volta). Il caso NP per piani corti segue dalla riduzione al SAT (ogni piano di lunghezza n può essere codificato come formula proposizionale di dimensione polinomiale in n e nella dimensione del dominio). ▫

19.2 Task and Motion Planning: Il Problema del Grounding Geometrico

Il task planning classico assume che le azioni siano applicabili indipendentemente dalla geometria. In robotica, invece, ogni azione simbolica (afferrare il bicchiere) richiede una verifica geometrica (il robot può raggiungere il bicchiere in questa configurazione? esiste una presa fattibile?) e la generazione di una traiettoria continua che la realizza.

Definizione 19.2 (Task and Motion Planning Problem). *Un problema TAMP è una tupla* $\langle D_{PDDL}, X, G, \{f_a\}_{a\in A}\rangle$ *dove: -* D_{PDDL} *è il dominio PDDL simbolico. -* X *è lo spazio geometrico continuo (configurazioni del robot e posizioni degli oggetti). -* $G: P\times O^* \to \{True, False, Unknown\}$ *è la funzione di ground truth geometrica che valuta i predicati sullo stato continuo. -* $f_a: X\to X$ *è la mappa di transizione geometrica dell'azione* a *(dalla cinematica del robot e delle azioni di manipolazione).*

La difficoltà fondamentale del TAMP è che i predicati PDDL devono essere **groundati** sullo spazio continuo: verificare se il predicato $On(cup, table)$ è vero richiede controllare se le coordinate continue del bicchiere sono compatibili con la superficie del tavolo, e i **parametri continui** delle azioni (la posa di grasp, il punto di deposito) devono essere trovati tramite pianificazione geometrica.

19.3 PDDLStream: Lazy Sampling e Ottimalità

PDDLStream (Garrett et al. 2020) è l'approccio attualmente più efficace al TAMP. L'idea chiave è di estendere il task planner con **stream**: generatori di valori continui che, dati dei parametri simbolici, producono candidati continui per le azioni.

Definizione 19.3 (Stream PDDLStream). *Uno stream è una tupla* $\langle I, O, \{(c_j, f_j)\}\rangle$ *dove* I *è un insieme di input simbolici/continui,* O *è un insieme di output continui, e* $\{(c_j, f_j)\}$ *è un insieme di coppie certificato-funzione: il certificato* c_j *è un predicato che viene reso vero quando la funzione generatrice* $f_j: I\to O$ *produce un output.*

Esempi di stream per manipolazione: - **IK stream**: dati oggetto e tipo di presa, genera configurazioni del robot q che soddisfano la cinematica inversa. Predicato: $Kin(obj, grasp, q)$. - **Grasp stream**: data la geometria dell'oggetto, genera pose di presa $g \in SE(3)$. Predicato: $Grasp(obj, g)$. - **Traj stream**: date configurazione iniziale e finale del robot, genera traiettorie collision-free ξ. Predicato: $Traj\left(q_{init}, q_{goal}, \xi\right)$.

L'algoritmo PDDLStream risolve il problema TAMP alternando due passi: (1) chiamata al planner simbolico su un problema PDDL esteso con i campioni finora generati dagli stream; (2) se il planner trova un piano candidato, verifica la fattibilità geometrica e, in caso di fallimento, campiona nuovi valori dagli stream per i passi falliti.

Teorema 19.2 (Completezza di PDDLStream, Garrett et al. 2020). *Se il problema TAMP ha soluzione e ogni stream ha probabilità positiva di generare soluzioni fattibili in ogni chiamata, allora PDDLStream restituisce una soluzione in tempo finito con probabilità 1.*

Dimostrazione. La completezza segue dall'analisi probabilistica: ad ogni iterazione, la probabilità di non trovare la soluzione è al più $\left(1 - p_{min}\right)^k$ con $p_{min} > 0$ la probabilità minima di ogni stream di generare una soluzione e k il numero di iterazioni. Per $k \to \infty$, questa probabilità tende a 0. La correttezza segue dalla verifica esplicita della fattibilità geometrica prima di restituire la soluzione. ▫

Connessione con la Pianificazione MPC del Capitolo 2

La connessione tra TAMP e MPC è concettuale ma profonda: entrambi pianificano su orizzonti finiti ottimizzando un obiettivo soggetto a vincoli. Il TAMP opera nel dominio simbolico-geometrico con orizzonte tipicamente di 5–20 azioni, ciascuna della durata di secondi; l'MPC del WBC opera nel dominio continuo con orizzonte di 0.5–1 secondi e passo di 10 ms. La connessione operativa è tramite il **semantic task interface**: il pianificatore TAMP genera una sequenza di pose di end-effector obiettivo $\{T_{goal}^{(1)}, T_{goal}^{(2)}, ...\}$ in SE(3), e ogni pose diventa un task del WBC con priorità alta.

19.4 LLM per la Generazione Automatica di Domini PDDL

La definizione manuale di domini PDDL richiede la formalizzazione esplicita di precondizioni e effetti di ogni azione, un processo laborioso e error-prone per domini robotici complessi. Gli LLM possono automatizzare questa generazione: dato un insieme di esempi di transizioni di stato osservate (o descritte in linguaggio naturale), un LLM addestrato genera automaticamente il dominio PDDL corrispondente. Questo approccio — denominato Domain Learning from Demonstrations (DLD) — riduce significativamente il costo di ingegneria della conoscenza.

Il framework LLM-PDDL

LLM-PDDL (Liu et al., 2024) usa GPT-4 come generatore di codice PDDL con un prompt strutturato che fornisce: (1) la descrizione del dominio in linguaggio naturale, (2) esempi di tracce di esecuzione (stato iniziale → azione → stato finale), e (3) un feedback strutturato sugli errori del planner. Il ciclo di generazione-validazione è iterativo:

D_PDDL ← LLM(task_description, traces, feedback_k)

plan_k ← PDDLPlanner(D_PDDL, s_0, g)

feedback_{k+1} ← Validator(plan_k, world_model)

La validazione usa un world model fisico (MuJoCo o Genesis) per verificare che ogni azione generata dal piano sia effettivamente eseguibile nell'ambiente fisico: se un'azione viola i vincoli geometrici o fisici, il feedback include la descrizione dell'errore e il planner viene reinvocato con il dominio corretto. Questo self-refinement loop converge in 3–7 iterazioni per la maggioranza dei task di manipolazione testati.

Completezza e correttezza del PDDL generato da LLM

Un problema fondamentale del LLM-PDDL è la verifica di correttezza del dominio generato. Un dominio PDDL è corretto se ogni azione ground soddisfa le sue precondizioni prima dell'esecuzione e produce gli effetti dichiarati dopo. La verifica formale richiede un model checker (come TLV

o SPIN) applicato all'asserzione di correttezza sul dominio, ma questo ha complessità PSPACE nel caso generale (Teorema 19.1).

Ai fini pratici, si adotta la verifica campionaria: si campionano K traiettorie casuali nel world model, si verifica che il piano PDDL le esegua correttamente, e si misura la fraction of successful executions. Per $K = 100$ e un bound di successo $\delta = 0.95$, con probabilità $\geq 1 - \delta$ il dominio ha accuracy $\geq 1 - \sqrt{(\ln(1/\delta)/(2K))} \approx 0.83$ per $\delta = 0.05$, $K = 100$ (bound di Hoeffding). Questo fornisce una garanzia statistica debole ma computazionalmente trattabile.

Capitolo 20 — LLM come Pianificatori Robotici: SayCan, Code as Policies e il Grounding Problem

20.1 Il Framework Unificato del Semantic Planning

L'uso di Large Language Models come pianificatori robotici si basa sull'osservazione che i modelli linguistici pre-addestrati su testi di internet contengono implicit knowledge sui compiti robotici: sanno che "fare il caffè" richiede prendere la tazzina, andare alla macchina, ecc. Il problema è il **grounding**: tradurre il piano linguistico in azioni fisiche eseguibili.

Definizione 20.1 (Semantic Planning Framework). *Un framework di semantic planning è una tupla* $\langle M_{LLM}, S_{skills}, G_{affordance}, F_{exec} \rangle$ *dove: -* $M_{LLM}: text \to P(text)$ *è il modello linguistico che genera distribuzioni su testo (piani, istruzioni). -* $S_{skills} = \{s_1, \ldots, s_k\}$ *è il vocabolario di skills eseguibili dal robot. -* $G_{affordance}: S_{skills} \times O_{obs} \to [0,1]$ *è la funzione di affordance che valuta la fattibilità di ogni skill nell'osservazione corrente. -* $F_{exec}: S_{skills} \to \pi$ *è il controller che esegue ogni skill.*

SayCan (Ahn et al. 2022) istanzia questo framework con:

$$a^* = arg\max_{a \in S} \underbrace{p(a|goal, context)}_{\text{LLM score}} \cdot \underbrace{G_{affordance}(a, o)}_{\text{feasibility score}}$$

Il punteggio LLM misura quanto l'azione a è linguisticamente plausibile nel contesto; il punteggio di affordance misura quanto l'azione è fisicamente eseguibile nell'osservazione corrente. La product of experts combina i due segnali.

Proposizione 20.1 (SayCan come pianificazione bayesiana). *Il punteggio SayCan è equivalente alla distribuzione a posteriori bayesiana su azioni quando il prior è il punteggio LLM e la likelihood è l'affordance:*

$$p(a|goal, o) \propto p_{LLM}(a|goal) \cdot p_{afford}(o|a)$$

dove $p_{afford}(o|a)$ *è la probabilità che l'osservazione* o *sia compatibile con l'esecuzione dell'azione* a.

20.2 Code as Policies: Grounding tramite Programmi Eseguibili

Code as Policies (Liang et al. 2023) usa un LLM per generare **codice Python** che realizza il piano robotico, invece di generare testo in linguaggio naturale. Il codice generato chiama una API di skill pre-implementate:

```
# Generato dall'LLM per "porta il bicchiere rosso al tavolo"
red_cup = find_object("red cup", camera_obs)
grasp_pose = compute_grasp(red_cup, grasp_type="top")
move_to(grasp_pose)
close_gripper()
target = find_object("table", camera_obs)
move_to(above(target))
open_gripper()
```

Il vantaggio di generare codice è che la struttura del programma (if-else, cicli, chiamate a funzione) impone una coerenza logica al piano che il linguaggio naturale non garantisce. Il problema del grounding si riduce all'implementazione corretta delle primitive (find_object, compute_grasp, ecc.).

Teorema 20.1 (Espressività di Code as Policies). *La classe di piani robotici esprimibili tramite programmi Python con accesso a un set di skill S è strettamente più ampia della classe di piani espressa da sequenze lineari di azioni SayCan:*

$$P_{CoP} \supsetneq P_{SayCan}$$

P_{CoP} include piani con branching condizionale (if-else su osservazioni), ripetizione (while/for su oggetti multipli), e composizione parametrica (chiamare la stessa skill con parametri diversi).

Dimostrazione. L'inclusione $P_{SayCan} \subseteq P_{CoP}$ è ovvia (ogni sequenza lineare è un programma Python senza branching). La stretta inclusione segue dall'esempio: il piano "per ogni tazza sul tavolo, portala nel lavandino" è esprimibile in CoP con un ciclo for cup in find_objects("cups"), ma non è

esprimibile come sequenza lineare in SayCan perché il numero di tazze non è noto a priori. ▫

20.3 Inner Monologue e Feedback Percettivo nel Loop di Pianificazione

Inner Monologue (Huang et al. 2022) chiude il loop di pianificazione includendo il **feedback percettivo** nel contesto del LLM: il robot esegue un'azione, osserva il risultato, e aggiorna il contesto dell'LLM con la descrizione dell'esito prima di pianificare l'azione successiva.

Il framework Inner Monologue può essere formalizzato come un **processo di decisione di Markov parzialmente osservabile** (POMDP) nel dominio linguistico:

$$s_{t+1} = world\left(s_t, a_t\right)$$

$$o_t = VLM\left(I_t\right)$$

(descrizione linguistica dell'osservazione)

$$a_t = LLM\left(o_0, a_0, o_1, a_1, \ldots, o_t, goal\right)$$

(piano condizionale sulla storia)

dove il Vision-Language Model (VLM) traduce le immagini del robot in descrizioni linguistiche dello stato, e l'LLM pianifica le azioni future condizionalmente all'intera storia linguistica.

Proposizione 20.2 (Convergenza del loop Inner Monologue). *Per un dominio di compiti con struttura finita (numero finito di stati simbolici e azioni), e assumendo che il VLM produca descrizioni corrette con probabilità* $p_{VLM} > 0$ *e che l'LLM selezioni azioni corrette con probabilità* $p_{LLM} > 0$*, il loop Inner Monologue completa il task in un numero atteso di passi* $E[T] \leq C/\left(p_{VLM} \cdot p_{LLM}\right)$ *dove* C *è il numero di stati nello spazio simbolico del dominio.*

20.4 Il Grounding Problem: Teoria e Connessione con l'Apprendimento per Trasferimento

Il **grounding problem** nel semantic planning è il problema di mappare simboli linguistici in concetti fisici: a cosa si riferisce "il bicchiere rosso"? dove si trova il "tavolo"? La soluzione standard è usare un **grounding model** $G: text \times I \rightarrow O$ che, dato un riferimento linguistico e un'immagine, produce un insieme di candidati oggetti fisici.

Proposizione 20.3 (Grounding come transfer learning). *Il problema del grounding è equivalente alla stima della distribuzione di oggetti* $p(o|text, I)$ *che può essere decomposta tramite Bayes:*

$$p(o|text, I) \propto p(text|o) \cdot p(o|I)$$

dove $p(o|I)$ *è il modello percettivo (rilevamento di oggetti) e* $p(text|o)$ *è il modello linguistico di riferimento. Questa decomposizione è esattamente la struttura di CLIP:* $p(text|o) \propto exp\left(\langle f_L(text), f_V\left(I_o\right)\rangle/\tau\right)$*, dove* I_o *è il crop dell'immagine relativo all'oggetto* o*.*

Parte VIII

Identificazione del Sistema, Calibrazione e Adattamento al Dominio

Capitolo 20.5 — Pianificazione Differenziabile e Calibrazione Targetless Camera-LiDAR

Raccordo con la monografia: Cap. 19 (TAMP, PDDLStream), Cap. 21.3 (calibrazione estrinseca).

6.1 SpatialVLM: Ragionamento Spaziale come Linguaggio

SpatialVLM (Chen et al., 2024) e un modello VLM addestrato esplicitamente per rispondere a domande spaziali quantitative sul mondo 3D: distanze, orientazioni, relazioni "sopra/sotto/sinistra", capacita di contenimento. Il training combina dati sintetici generati da simulatori 3D (centinaia di milioni di coppie domanda-risposta) con annotazioni reali sparse.

La capacita di ragionamento spaziale di SpatialVLM abilita un nuovo paradigma di TAMP: invece di specificare manualmente le precondizioni e gli effetti delle azioni in PDDL (Cap. 19), il planner usa SpatialVLM per valutare la fattibilita geometrica tramite domande in linguaggio naturale. Ad esempio, la precondizione Reachable(gripper, cup) e valutata chiedendo a SpatialVLM "La tazza e raggiungibile dal gripper nella configurazione corrente?", ricevendo una risposta si/no con confidenza.

6.2 Pianificazione Differenziabile: Ottimizzazione del Piano tramite Gradiente

Il Task and Motion Planning classico (Cap. 19) risolve un problema combinatorio discreto (trovare la sequenza di azioni simboliche) separato da un problema continuo (trovare i parametri geometrici). La pianificazione differenziabile unifica i due livelli: il piano viene rappresentato come una sequenza di parametri continui theta = (theta_1, ..., theta_N) -- pose di end-effector, configurazioni intermediate, parametri di presa -- e ottimizzato congiuntamente tramite gradient descent attraverso il simulatore differenziabile:

theta* = argmin_{theta} L_task(execute(theta)) + lambda * L_smooth(theta)

dove execute(theta) e il simulatore differenziabile (MuJoCo MJX o Genesis, Cap. 9) che propaga il piano fino all'output, L_task misura il fallimento del task (es. distanza dello stato finale dal goal), e L_smooth penalizza la non-smoothness delle traiettorie. Il gradiente nabla_{theta} L e calcolato tramite reverse-mode autodiff attraverso il simulatore (eq. 9.1, differenziazione implicita del LCP).

Proposizione S6.1 (Convergenza locale della pianificazione differenziabile).

Per un simulatore C^2-regolare (senza eventi di contatto nel piano ottimale) e con L_task fortemente convessa nella regione feasible, il gradient descent sulla pianificazione differenziabile con passo alpha <= 1/L converge a un minimo locale con tasso lineare 1 - alpha * mu, dove mu e la costante di convessita di L_task e L e la sua costante di Lipschitz del gradiente.

6.3 Calibrazione Estrinseca Targetless Camera-LiDAR

La calibrazione estrinseca tra camera e LiDAR (sezione 21.3 della monografia) e tradizionalmente effettuata tramite un target calibrato (scacchiera o target riflettente). I metodi targetless eliminano la necessita di target specifici, sfruttando invece le feature naturali della scena.

Il metodo piu efficace per la calibrazione targetless camera-LiDAR e basato sulla consistenza fotometrica: le feature della nuvola di punti LiDAR (es. bordi di edifici, angoli) vengono proiettate sull'immagine usando la trasformazione estrinseca T_CL stimata, e la consistenza fotometrica (similarita tra l'intensita attesa e quella misurata) viene massimizzata:

> T_CL* = argmax_{T in SE(3)} (1/N) sum_{i=1}^{N} MI(I(pi(T * p_i)), Intensity_LiDAR(p_i))

dove MI e la mutual information, pi e la proiezione prospettica, e p_i sono i punti LiDAR. L'ottimizzazione e eseguita con il metodo di Powell su SE(3), che evita le singolarita di parametrizzazione degli angoli di Eulero.

Teorema S6.1 (Identificabilita della calibrazione targetless).

La calibrazione targetless camera-LiDAR e identificabile se e solo se la scena ha sufficiente struttura 3D: almeno tre piani non complanari devono essere visibili simultaneamente nella camera e nel LiDAR. In assenza di questa condizione (es. un corridoio rettilineo), i parametri di traslazione lungo l'asse di simmetria della scena non sono identificabili.

Dimostrazione. L'analisi delle singolarita della matrice di Fisher dell'obiettivo di mutual information rivela che le direzioni di informazione nulla corrispondono esattamente alle simmetrie della scena. Per scene con tre piani non complanari, tutte le direzioni di SE(3) sono eccitate e la matrice di Fisher e invertibile. La prova formale usa il teorema di identificabilita della regressione non lineare (Rothenberg 1971) applicato al modello di proiezione prospettica. []

Parte VIII — Stima dei Parametri, Calibrazione Multi-Sensore e Sim-to-Real Transfer

Tutti i modelli matematici della monografia — dalle equazioni del moto lagrangiane del Capitolo 1 ai world models del Capitolo 8 — richiedono parametri numerici: masse, inerzie, attrito, geometrie dei link, calibrazione dei sensori. Questi parametri non sono mai noti a priori con precisione sufficiente: il robot reale differisce dal suo modello CAD per tolleranze di produzione, deformazioni elastiche dei link, degrado delle superfici di contatto, e incertezza nei guadagni degli attuatori. La **system identification** è la teoria e la pratica della stima di questi parametri a partire da misurazioni, e il **domain adaptation** è l'insieme delle tecniche per ridurre il gap tra simulazione e realtà.

Capitolo 21 — System Identification per Robot Umanoidi

21.1 Identificabilità dei Parametri Inerziali: la Matrice di Regressione

Per un sistema multi-corpo con N_b link, i parametri inerziali di ogni link sono: massa m_i, primo momento di massa $m_i c_i \in R^3$ (prodotto di massa per posizione del CoM nel frame del link), e tensore di inerzia $I_i^c \in R^{3\times3}$ rispetto al CoM (matrice simmetrica, 6 parametri indipendenti). Il totale è $10N_b$ parametri inerziali.

Il punto di partenza dell'identificazione è la forma **lineare nei parametri** delle equazioni del moto lagrangiane. Definendo il vettore dei parametri inerziali $\pi \in R^{10N_b}$ e il **regressore** $Y\left(\eta, \dot{\eta}, \ddot{\eta}\right) \in R^{n\times10N_b}$:

$$\tau = Y\left(\eta, \dot{\eta}, \ddot{\eta}\right)\pi$$

questa relazione è lineare in π per ogni configurazione e accelerazione del robot. Il regressore Y si calcola esplicitamente tramite l'algoritmo di Featherstone modificato.

Teorema 21.1 (Kawasaki 1991 — Identificabilità dei parametri inerziali). *I parametri inerziali* π *sono identificabili tramite misure di coppia* τ *se e solo se la matrice di regressione accumulata* $W = \sum_{k=1}^{K} Y^{\top}\left(t_k\right)Y\left(t_k\right)$ *ha rango pieno* $p_{id} = rank(W)$. *Il numero* p_{id} *è il numero di* ***parametri base*** *(base inertial parameters), l'insieme minimale di combinazioni lineari di* π *che compaiono nelle equazioni del moto.*

Dimostrazione. La linearità della relazione $\tau = Y\pi$ implica che con K misure, il problema di stima è il sistema lineare $\left[Y_1; ...; Y_K\right]\pi = \left[\tau_1; ...; \tau_K\right]$. Il vettore π è identificabile (unico) se e solo se $W = \sum_k Y_k^{\top} Y_k$ è invertibile, ovvero ha rango pieno. Se $rank(W) < 10N_b$, il null-space di W contiene

combinazioni di π non eccitabili: si ricava solo la proiezione di π sullo spazio delle colonne di W, i parametri base. ▫

Traiettorie Persistentemente Eccitanti

Il rango di W dipende dalla ricchezza delle traiettorie di eccitazione eseguite durante l'identificazione. Una traiettoria $\eta(t)$ è **persistentemente eccitante** (PE) per l'identificazione se $rank(W) = p_{id}$ per ogni T sufficientemente lungo. Le traiettorie ottimali per l'identificazione minimizzano il numero di condizione di W (per minimizzare l'amplificazione del rumore sulle stime):

$$\eta_{PE}^{*} = arg \min_{\eta(\cdot)} \kappa(W) = argmin \frac{\lambda_{max}(W)}{\lambda_{min}(W)}$$

Questo problema si risolve tipicamente con traiettorie parametrizzate come somme di N_h armoniche:

$$q_k(t) = \sum_{j=1}^{N_h} \left[a_{kj} sin\left(j\omega_0 t\right) + b_{kj} cos\left(j\omega_0 t\right) \right]$$

con i coefficienti $\{a_{kj}, b_{kj}\}$ ottimizzati rispetto a $\kappa(W)$ tramite ottimizzazione non lineare.

21.2 Stima ai Minimi Quadrati con Rumore di Misura

Dato un dataset di K misure $\left(Y_k, \tau_k\right)$ con rumore $\tau_k = Y_k \pi + \varepsilon_k$ con $\varepsilon_k \sim N\left(0, \Sigma_\tau\right)$, lo stimatore a **minimi quadrati pesati** (WLS) è:

$$\hat{\pi}_{WLS} = \left(\sum_k Y_k^\top \Sigma_\tau^{-1} Y_k \right)^{-1} \sum_k Y_k^\top \Sigma_\tau^{-1} \tau_k = W_\Sigma^{-1} b_\Sigma$$

Lo stimatore WLS è il **Best Linear Unbiased Estimator** (BLUE) per π:

Teorema 21.2 (Gauss-Markov per l'identificazione robotica). *Lo stimatore WLS $\hat{\pi}_{WLS}$ è non distorto ($E\left[\hat{\pi}\right] = \pi$) e ha varianza minima tra tutti gli stimatori lineari non distorti di* π. *La sua matrice di covarianza è:*

$$Cov\left(\hat{\pi}_{WLS}\right) = W_{\Sigma}^{-1} = \left(\sum_{k} Y_k^{\top} \Sigma_{\tau}^{-1} Y_k\right)^{-1}$$

Dimostrazione. Il BLUE per il modello lineare $\tau = Y\pi + \varepsilon$ con $\varepsilon \sim N\left(0, \Sigma_{\tau}\right)$ segue direttamente dal teorema di Gauss-Markov: lo stimatore WLS minimizza $\left(\hat{\pi} - \pi\right)^{\top} A \left(\hat{\pi} - \pi\right)$ per ogni matrice di peso positiva definita A. La covarianza si calcola propagando la covarianza del rumore attraverso l'espressione dello stimatore. ▫

Stima Robusta con M-Estimator

In presenza di **outlier** (misurazioni corrotte da impatti improvvisi o guasti dei sensori), lo stimatore WLS amplifica l'errore perché minimizza i residui quadratici. Gli **M-estimator** sostituiscono la funzione di costo quadratica con una funzione robusta ρ:

$$\hat{\pi}_M = arg \min_{\pi} \sum_{k} \rho\left(\frac{\tau_k - Y_k \pi}{\sigma}\right)$$

La funzione di Huber $\rho_{Huber}(r) = r^2/2$ per $|r| \leq c$ e $c|r| - c^2/2$ per $|r| > c$ (con c scelto da tabelle di robustezza) ha derivata $\psi_{Huber}(r) = rmin(1, c/|r|)$ che non amplifica i residui grandi. La soluzione si ottiene con l'algoritmo **Iteratively Reweighted Least Squares** (IRLS):

$$\hat{\pi}^{(l+1)} = \left(Y^{\top} W^{(l)} Y\right)^{-1} Y^{\top} W^{(l)} \tau, \quad W_{kk}^{(l)} = \psi_{Huber}\left(r_k^{(l)}\right) / r_k^{(l)}$$

dove $r_k^{(l)} = \tau_k - Y_k \hat{\pi}^{(l)}$ sono i residui all'iterazione l.

21.3 Calibrazione Estrinseca Multi-Sensore: Camera-IMU-LiDAR

Un robot umanoide moderno è equipaggiato con almeno una IMU (accelerometro + giroscopio), una o più camere, e possibilmente un LiDAR. La **calibrazione estrinseca** stima le pose relative tra i sensori — le trasformazioni $T_{cam}^{IMU}, T_{LiDAR}^{cam} \in SE(3)$ — senza le quali la fusione sensoriale è incoerente.

Camera-IMU: Calibrazione tramite Moto Eccitante

Il problema di calibrazione camera-IMU si formula come l'ottimizzazione non lineare sulla trasformazione $T = (R, t) \in SE(3)$:

$$T^{*} = arg \min_{T \in SE(3)} \sum_{k=1}^{K} \| \omega_{cam}^{(k)} - R\omega_{IMU}^{(k)} \|_{\Omega}^{2} + \| a_{cam}^{(k)} - Ra_{IMU}^{(k)} - t \times \dot{\omega}_{cam}^{(k)} \|_{A}^{2}$$

dove $\omega_{cam}^{(k)}$ e $a_{cam}^{(k)}$ sono le velocità angolari e le accelerazioni stimate dalla camera (tramite stima del moto visivo), $\omega_{IMU}^{(k)}$ e $a_{IMU}^{(k)}$ le corrispondenti misure IMU, e Ω, A le matrici di pesatura inversamente proporzionali alle varianze di misura.

Teorema 21.3 (Identificabilità della calibrazione camera-IMU). *La calibrazione camera-IMU è identificabile (unica soluzione) se e solo se la traiettoria eseguita durante la calibrazione include almeno tre assi di rotazione linearmente indipendenti e almeno un tratto con accelerazione non nulla non allineata alla gravità.*

Dimostrazione. Il termine rotazionale identifica R se le velocità angolari $\omega_{IMU}^{(k)}$ spannano R^3 (tre assi indipendenti). Il termine traslazionale identifica t se le accelerazioni $a_{cam}^{(k)}$ hanno una componente non allineata con g (la componente di gravità è costante e non contribuisce all'identificazione di t). La condizione di tre assi indipendenti garantisce il rango pieno della matrice di regressione per R; la condizione sull'accelerazione garantisce l'identificabilità di t. ▫

21.4 Identificazione dei Parametri di Flessibilità Strutturale

I modelli di corpo rigido del Capitolo 1 assumono link indeformabili, un'approssimazione valida per robot compatti ma sempre meno accurata per robot umanoidi con struttura leggera ottimizzata per il peso. I link del torso, delle gambe e delle braccia lunghe presentano flessibilità elastica significativa — in particolare le braccia di Figure 02 con SEA (§1.1) mostrano risonanze strutturali nell'intervallo 5–30 Hz — che devono

essere incluse nel modello per evitare eccitazione delle mode elastiche durante il controllo ad alto guadagno.

Il modello Flexible-Link Robot (FLR)

Un link flessibile è modellato tramite la decomposizione modale: le deformazioni elastiche u(x,t) del link come campo continuo sono proiettate su un numero finito di modi normali {φ◎(x)}: u(x,t) = Σ◎ q◎(t) φ◎(x) dove q◎(t) sono le coordinate modali. Le equazioni del moto del robot flessibile diventano un sistema accresciuto con le coordinate modali come gradi di libertà aggiuntivi:

$$M_{aug}(\eta, q_f)\,\ddot{\eta}_{aug} + C_{aug}(\eta, q_f, \dot{\eta}_{aug})\,\dot{\eta}_{aug} + K_f\,q_f = S_{aug}\,\tau$$

dove η_aug = (η, q_f)^T, K_f = diag(K◎) è la matrice di rigidezza modale e K◎ = ω◎² M◎ con ω◎ la k-esima frequenza naturale e M◎ la massa modale. L'identificazione dei parametri modali (ω◎, M◎, coefficiente di smorzamento ζ◎) si effettua tramite Experimental Modal Analysis (EMA): si applica una forza impulsiva al link e si misura la risposta in accelerazione, poi si esegue la decomposizione nel dominio della frequenza tramite l'algoritmo PolyMAX o LSCE.

Integrazione con il WBC: filtraggio modale

Una volta identificati i modi elastici, si possono applicare due strategie nel WBC: il filtraggio input-shaping e l'active vibration control. L'input-shaping filtra il segnale di riferimento con una sequenza di impulsi calibrata per cancellare l'eccitazione della risonanza. Per una risonanza a frequenza ω◎ con smorzamento ζ, il filtro ZV (Zero Vibration) è: ZV_filter = [1, K/(1+K)] * [1, 0] + [0, K/(1+K)] * [0, 1/2] K = exp(−π ζ / √(1−ζ²)), T_d = π / (ω◎ √(1−ζ²))

Il filtro ZV introduce un ritardo di T_d/2 nel segnale di riferimento, che per ω◎ = 20 Hz e ζ = 0.1 è T_d/2 ≈ 25 ms — un ritardo accettabile per il controllo di postura ma critico per riflessi rapidi. In questi casi si preferisce l'active vibration control: aggiungere coppie di smorzamento τ_vib = −D q̇_f alle coppie del WBC, dove D è una matrice di smorzamento attivo stimata dall'output dei sensori di deformazione nei link.

Capitolo 21.5 — Reti Neurali Informate dalla Fisica (PINN) per la Stima Online dei Parametri Inerziali

Raccordo con la monografia: § 21.1–21.2 (system identification, regressore Y, stimatore WLS).

Il framework classico di system identification (Cap. 21) ipotizza che i parametri pi siano costanti e che il dataset di identificazione sia raccolto offline con traiettorie persistentemente eccitanti. Entrambe le ipotesi vengono meno nei robot che operano in ambienti non strutturati: il payload varia (un robot che trasporta oggetti di massa ignota) e il tempo disponibile per eccitare tutti i modi cinematici è limitato. Le Physics-Informed Neural Networks (PINN) offrono un approccio alternativo che incorpora le equazioni del moto come vincolo di regolarizzazione nella loss della rete, consentendo la stima online dei parametri anche in condizioni di eccitazione parziale.

1.1 Architettura PINN per la Dinamica dei Robot

Sia la rete f_theta: (eta, eta_dot, tau) -> pi_hat un approssimatore dei parametri inerziali, parametrizzato da theta. La PINN viene addestrata minimizzando una loss composita:

L_PINN(theta) = L_data(theta) + lambda_phys * L_phys(theta) + lambda_reg * L_reg(theta)

dove:

L_data(theta) = (1/K) sum_{k=1}^{K} || tau_k - Y(eta_k, eta_dot_k, eta_ddot_k) f_theta(eta_k, eta_dot_k, tau_k) ||^2

L_phys(theta) = || M(f_theta) - M(f_theta)^T ||_F^2 + max(0, -lambda_min(M(f_theta)))

L_reg(theta) = || f_theta - pi_nom ||^2

Il termine L_phys penalizza le violazioni della simmetria e della definitezza positiva della matrice di inerzia M(pi_hat), che sono proprieta fisiche hard che la rete altrimenti potrebbe violare. Il termine L_reg agisce come prior gaussiano attorno ai valori nominali pi_nom del CAD.

1.2 Stima Online con Sliding Window

Per la stima in tempo reale, si adotta una finestra scorrevole di W passi temporali. Ad ogni ciclo di controllo t, si aggiorna theta tramite un passo di gradient descent sulla finestra [t-W, t]:

theta_{t+1} = theta_t - alpha * nabla_theta L_PINN(theta_t; D_{t-W:t})

La scelta di W governa un tradeoff tra reattivita (W piccolo: theta si adatta rapidamente ai cambiamenti del payload) e varianza della stima (W grande:

theta e piu stabile ma lento ad adattarsi). Per un robot che solleva oggetti di massa variabile nell'intervallo [0, 20] kg, risultati sperimentali su Franka Panda indicano W = 200 ms ottimale con alpha = 1e-4.

1.3 Garanzie di Consistenza: Teorema di Convergenza PINN

Teorema S1.1 (Consistenza dello stimatore PINN).

Sia f_theta* il minimizzatore globale di L_PINN con lambda_phys -> infty. Allora f_theta*(eta, eta_dot, tau) e il Best Linear Unbiased Estimator (BLUE) dei parametri identificabili nell'insieme C_phys = {pi : M(pi) >= 0, M(pi) = M(pi)^T}, ovvero la proiezione dello stimatore WLS (Teorema 21.2) sul sottinsieme fisicamente ammissibile.

Dimostrazione. Il limite lambda_phys -> infty forza la soluzione nel sottinsieme C_phys tramite il metodo della penalizzazione esterna. Sul sottinsieme C_phys, il termine L_data coincide con la loss WLS, e lo stimatore risultante eredita le proprieta BLUE dal Teorema di Gauss-Markov (Teorema 21.2). La proiezione su C_phys non introduce distorsione se pi_reale appartiene all'interno di C_phys, condizione sempre verificata per sistemi fisici. []

1.4 Confronto PINN vs. EKF per la Stima Adattiva del Payload

Il filtro di Kalman esteso (EKF, §22.4) e la PINN online rappresentano due approcci complementari alla stima adattiva. L'EKF e ottimale nel senso bayesiano per sistemi lineari con rumore gaussiano, ma richiede la linearizzazione del modello di contatto h_contact(phi, x), che introduce errori sistematici per modelli di contatto altamente non lineari. La PINN, al contrario, approssima universalmente la relazione tau = Y(eta, eta_dot, eta_ddot) pi senza linearizzazione, a costo di un maggiore onere computazionale (un passo di gradient vs. la ricorrenza di Kalman).

Per payload variabili e modelli di contatto non lineari, la PINN produce errori di stima inferiori del 30-40% rispetto all'EKF sulla norma || pi_hat - pi_reale ||, ma con latenza di aggiornamento 5-10x superiore. L'integrazione dei due approcci -- PINN per l'inizializzazione periodica e EKF per il tracking real-time -- e la soluzione pratica adottata nei sistemi piu avanzati.

Capitolo 22 — Domain Adaptation e Sim-to-Real Transfer

22.1 Il Sim-to-Real Gap: Tassonomia delle Fonti di Discrepanza

Il gap tra simulazione e realtà per i robot umanoidi si articola in quattro categorie:

1. Parametri fisici errati: masse, inerzie, attrito, smorzamento non coincidono con i valori reali. Questo genera discrepanze nelle forze e nei momenti calcolati dal simulatore.

2. Contatti non modellati: la simulazione di contatti morbidi (deformazione del piede, elasticità del terreno) con modelli rigidi introduce errori sistematici nella dinamica della locomozione.

3. Ritardi e latenze attuative: il simulatore tipicamente assume controllo senza ritardo; il robot reale ha ritardi di 1–10 ms nella catena attuatori che possono destabilizzare i controllori ad alto guadagno.

4. Gap percettivo (visual sim-to-real): le immagini generate dai simulatori differiscono dalle immagini reali per textures, illuminazione, riflessi, e rumore del sensore. Questo è il gap più rilevante per i modelli VLA.

22.2 Domain Randomization come Distribuzione a Priori Robusta

La domain randomization (introdotta nella sezione 9.5 del Completamento della Parte III) si specializza per i robot umanoidi con le seguenti distribuzioni sui parametri fisici:

$$\phi = \Big(\underbrace{m_1, \ldots, m_{N_b}}_{masse}, \underbrace{vec\left(I_1^c\right), \ldots, vec\left(I_{N_b}^c\right)}_{tensori\ inerzia}, \underbrace{\mu_1, \ldots, \mu_k}_{attrito\ giunti}, \underbrace{\Delta\tau_1, \ldots, \Delta\tau_n}_{ritardi\ attu}$$

La distribuzione tipica è $\phi \sim \prod_i U\left[\phi_i^{nom}(1-\delta_i), \phi_i^{nom}(1+\delta_i)\right]$ con $\delta_i \in [0.1, 0.5]$ (10–50% di variazione intorno al valore nominale). I ritardi $\Delta\tau_k \sim U[0, \Delta\tau_{max}]$ con $\Delta\tau_{max} =$ 1–5 ms.

Proposizione 22.1 (Ottimalità della prior uniforme per domain randomization). *Per parametri ϕ con prior uniforme $U[a,b]$ e funzione di valore localmente convessa V_ϕ^π in ϕ, la policy $\pi_{DR}^* = argmax_\pi E_\phi\left[V_\phi^\pi\right]$ soddisfa:*

$$V_{DR}^\pi(\phi_{real}) \geq V_{nom}^\pi(\phi_{real}) - \frac{1}{2} \| \nabla_\phi^2 V_{nom}^\pi \|_2 \cdot Var(\phi)$$

ovvero la policy con domain randomization ha valore reale al più $O(Var(\phi))$ inferiore alla policy nominale ottimale.

Dimostrazione. Espandendo V_ϕ^π al secondo ordine intorno a ϕ_{nom} e prendendo il valore atteso rispetto alla prior uniforme: $E_\phi\left[V_\phi^\pi\right] = V_{nom}^\pi + \frac{1}{2} tr\left(\nabla_\phi^2 V^\pi \cdot Cov(\phi)\right) + O\left(Var^{3/2}\right)$. Il termine quadratico è negativo per V^π localmente concava in ϕ (convessità della loss). La policy DR massimizza questa quantità, che è una lower bound sul valore reale a ϕ_{real} in un intorno dell'ipotesi $\phi_{real} \approx \phi_{nom}$. ▫

22.3 Domain Adaptation Neurale: DANN e la Teoria di Ben-David

Quando la domain randomization non è sufficiente — perché il gap visivo è troppo grande o perché i parametri rilevanti non sono randomizzabili — si usa il **domain adaptation** neurale. L'approccio più consolidato è il **Domain-Adversarial Neural Network** (DANN, Ganin et al. 2016).

In DANN, il modello è decomposto in un **feature extractor** $G_f: X \to F$, un **task classifier** $G_y: F \to Y$, e un **domain discriminator** $G_d: F \to \{0, 1\}$ (0 = simulazione, 1 = reale). L'obiettivo è:

$$\min_{G_f, G_y} \max_{G_d} E_{x \sim D_{sim}}\left[L_y\left(G_y\left(G_f(x)\right), y\right)\right] - \lambda E_{x \sim D_{sim} \cup D_{real}}\left[L_d\left(G_d\left(G_f(x)\right), d\right)\right]$$

Il termine di massimizzazione del discriminatore forza G_f a produrre feature non distinguibili tra simulazione e realtà — riducendo il termine $d_{H\Delta H}$ della bound di Ben-David (Teorema 7.2 del Completamento della Parte II).

Teorema 22.1 (Convergenza DANN con bound di generalizzazione). *Se il feature extractor G_f^* è il minimizzatore dell'obiettivo DANN, allora il bound di generalizzazione sul dominio target (reale) è:*

$$\varepsilon_{real}\left(G_y \circ G_f^*\right) \leq \varepsilon_{sim}\left(G_y \circ G_f^*\right) + d_{H\Delta H}^{DANN}\left(p_{sim}, p_{real}\right) + \lambda^*$$

dove $d_{H\Delta H}^{DANN}$ è la distanza di dominio nell'iperspazio delle feature di G_f^*, che per il DANN ben addestrato soddisfa $d_{H\Delta H}^{DANN}\left(p_{sim}, p_{real}\right) \leq d_{H\Delta H}^{input}\left(p_{sim}, p_{real}\right)$.

Dimostrazione. Il bound di generalizzazione segue direttamente dal Teorema 7.2 applicato allo spazio delle feature F invece dello spazio degli input. La disuguaglianza $d_{H\Delta H}^{DANN} \leq d_{H\Delta H}^{input}$ segue dal fatto che l'ottimizzazione avversariale trova il feature extractor che minimizza la distanza di dominio nello spazio delle feature — il mapping G_f^* è quindi una proiezione che non può aumentare la distanza di dominio rispetto allo spazio degli input. ▫

22.4 Sim-to-Real per Policy Locomotorie: Struttura dei Residui e Apprendimento Adattivo

Per le policy di locomozione bipede, il sim-to-real transfer presenta sfide specifiche legate alla discontinuità dei contatti. La tecnica più efficace è il **residual policy learning** (Lee et al. 2020, Kumar et al. 2021): si mantiene un controller base π_{base} derivato dalla simulazione, e si addestra una policy residua π_{res} che corregge l'azione per l'errore del simulatore:

$$u_{real} = \pi_{base}(x) + \pi_{res}\left(x; \phi_{est}\right)$$

dove ϕ_{est} è una stima a runtime dei parametri fisici del terreno (che tipicamente non sono disponibili a priori). La policy π_{res} viene addestrata con dati reali o con fine-tuning in simulazione su errori simulati.

Proposizione 22.2 (Errore ottimale del residual policy learning). *Sia* $\Delta_\phi = \pi^*_{real}(x) - \pi_{base}(x)$ *la correzione ottimale rispetto ai parametri reali* ϕ_{real}. *Il residual learning con funzione approssimante nella classe H raggiunge errore:*

$$\| u_{real} - \pi^*_{real}(x)\|^2 = \|\pi_{res}(x) - \Delta_\phi(x)\|^2 \leq inf_{h\in H}\|h - \Delta_\phi\|^2 + \varepsilon_{stat}$$

dove il primo termine è l'errore di approssimazione (quanto H può approssimare la correzione ideale) e ε_{stat} *è l'errore statistico (quanto bene si stima* $h\in H$ *con dati finiti).*

Dimostrazione. La decomposizione segue dalla decomposizione bias-varianza classica. L'errore del residual learning è bounded dalla somma dell'errore di approssimazione nella classe *H* (bias) e dell'errore di stima di π_{res} nell'insieme H con dati reali finiti (varianza). La seconda parte è bounded dall'analisi standard della regressione tramite Rademacher complexity di H. ▫

Stima Adattiva dei Parametri del Terreno a Runtime

Un aspetto cruciale per la locomozione su terreni non strutturati è la **stima in tempo reale** dei parametri del terreno $\phi_{terrain}$ (durezza, attrito, geometria). Questo si realizza con un **filtro di Kalman esteso** (EKF) applicato al modello della dinamica del contatto:

$$\phi_{t+1} = \phi_t + w_t, \quad w_t \sim N(0, Q)$$

$$z_t = h_{contact}(\phi_t, x_t) + v_t, \quad v_t \sim N(0, R)$$

dove z_t è la misura della forza di contatto e $h_{contact}$ è il modello del contatto elastico. L'EKF aggiorna la stima $\hat{\phi}_t$ ad ogni ciclo di controllo:

$$\hat{\phi}_{t|t} = \hat{\phi}_{t|t-1} + K_t\left(z_t - h_{contact}\left(\hat{\phi}_{t|t-1}, x_t\right)\right)$$

$$K_t = P_{t|t-1} H_t^\top \left(H_t P_{t|t-1} H_t^\top + R \right)^{-1}$$

dove $H_t = \partial h_{contact} / \partial \phi |_{\hat{\phi}_{t|t-1}}$ è lo Jacobiano del modello di contatto e $P_{t|t-1}$ è la matrice di covarianza dell'errore di stima. La stima adattiva $\hat{\phi}_t$ viene poi usata nella policy residua $\pi_{res}\left(x; \hat{\phi}_t\right)$ per correggere l'azione del controller base in funzione delle condizioni di terreno stimate.

22.5 Meta-Learning per Adattamento Rapido al Dominio Reale

Il sim-to-real transfer classico (§22) adatta la policy al dominio reale tramite fine-tuning su pochi dati reali, il che richiede comunque centinaia o migliaia di rollout nel robot reale. Il meta-learning offre un approccio alternativo: invece di adattare la policy a un singolo dominio reale, si addestra la policy su una distribuzione di domini simulati in modo che l'adattamento a un nuovo dominio (il mondo reale) richieda solo pochi step di gradient.

MAML applicato alla robotica

Model-Agnostic Meta-Learning (MAML, Finn et al., 2017) addestra i parametri θ di una policy neurale su un set di task $\{\mathcal{T}_i\}$ (diversi simulatori con parametri randomizzati) minimizzando la loss post-adattamento:

$$L\_MAML(\theta) = \Sigma_i \, L\_\mathcal{T}_i(\theta - \alpha \, \nabla\_\theta \, L\_\mathcal{T}_i(\theta))$$

Il gradiente del meta-obiettivo ∂L_MAML/∂θ richiede la derivata seconda attraverso un passo di gradient descent — denominata meta-gradient — calcolabile tramite reverse-mode autodiff di ordine 2 (Hessian-vector product). Il costo computazionale è O(n_θ) per l'Hessian-vector product tramite l'approssimazione di Pearlmutter, rendendolo scalabile a modelli di policy con milioni di parametri.

Per la robotica, MAML è applicato sulla distribuzione dei parametri fisici del simulatore: ogni task $\mathcal{T}_i$ corrisponde a un simulatore con parametri fisici campionati da P_rand. Dopo il meta-training, l'adattamento al robot

reale richiede solo K = 10–50 rollout nel mondo reale per identificare i parametri fisici rilevanti tramite K passi di gradient:

$$\theta\_real = \theta - \alpha \nabla\_\theta \Sigma_{i=1}^{K} L\_real(\tau_i, \theta)$$

Il risultato empirico — riportato da Nagabandi et al. (2019) per sistemi locomotori e da Yu et al. (2020) per manipolazione — è che MAML richiede 3–5× meno dati reali rispetto al fine-tuning standard per ottenere la stessa performance, grazie alla meta-inizializzazione che facilita l'adattamento.

Connessione con la teoria di Bayes e bound di adattamento

Il meta-learning può essere interpretato come stima bayesiana: la meta-policy θ è la prior $P(\theta)$, e l'adattamento $\theta\_real = \theta - \alpha \nabla L\_real$ è un passo di gradient verso la posterior $P(\theta \mid D\_real) \propto L(D\_real \mid \theta) P(\theta)$. La qualità dell'adattamento è misurata dal bound di generalizzazione:

$$E_{\mathscr{T}\_real}[L(\theta\_real)] \leq E_{\mathscr{T}\_sim}[L(\theta\_train)] + C \cdot \sqrt{(D\_KL(P\_real \,\|\, P\_train) / K)}$$

dove C dipende dalla curvatura della loss nella direzione di adattamento e K è il numero di rollout reali. Il bound mostra che meta-learning è vantaggioso rispetto al fine-tuning quando la divergenza KL tra dominio reale e simulato è bassa — esattamente quando la domain randomization è efficace.

22.6 Architetture Neurosimboliche: Integrazione di Ragionamento Formale e Apprendimento

Le architetture neurosimboliche (NeSy) mirano a integrare i punti di forza del ragionamento simbolico — correttezza logica, generalizzazione sistematica, interpretabilità — con quelli dell'apprendimento neurale — flessibilità, tolleranza al rumore, scalabilità con i dati. Per la robotica, questa integrazione è motivata dalla necessità di sistemi che possano sia manipolare oggetti in modo sicuro e certificato (proprietà simboliche) sia adattarsi a ambienti non strutturati (proprietà neurali).

22.6.1 Il Framework DeepProbLog e le Regole Probabilistiche

DeepProbLog (Manhaeve et al., 2018) estende la logica probabilistica Problog con reti neurali come funzioni di probabilità. Una regola DeepProbLog ha la forma:

P :: head(X) :- body(X, nn(f_θ, X, Y))

dove P è una probabilità, head e body sono letterali logici, e nn(f_θ, X, Y) è un letterale neuronale che chiama la rete f_θ con input X e produce output Y come distribuzione di probabilità. Il sistema integra la retropropagazione attraverso le reti neurali con il ragionamento probabilistico forward-backward sulla struttura logica.

Per la robotica, le regole DeepProbLog possono esprimere constraint di sicurezza come: safe_action(Action, State) :- not(violates_cbf(State, Action)), P(Action | State) ≥ p_min. Il termine violates_cbf è verificato simbolicamente tramite la condizione CBF (§16.2), mentre P(Action | State) è calcolato dalla policy neurale π_θ. L'integrazione produce una policy ibrida che è safety-certified per costruzione e robusta grazie alla rete neurale.

22.6.2 Programmazione Logica per il Grounding Robotico

Il grounding problem (§20.4) si presta naturalmente a un trattamento neurosimbolico: il programma logico definisce le relazioni simboliche tra oggetti (on(A,B), near(A,B), grasped(gripper, A)), mentre le reti neurali stimano le probabilità di questi predicati dalle osservazioni visive. Il sistema Answer Set Programming (ASP) — con solver come Clingo — è particolarmente adatto per il ragionamento sotto constraint: dato un insieme di fatti (stimati neuralmente) e un programma di regole, ASP trova tutti i modelli stabili (answer sets) che soddisfano le regole.

Per il task planning robotico, un answer set corrisponde a un piano valido: una sequenza di azioni ground che soddisfa le precondizioni (verificate simbolicamente) con gli effetti stimati neuralmente. La complessità di Clingo per trovare un answer set è NP-hard nel caso peggiore (come PDDL, Teorema 19.1), ma per programmi con struttura stratificata — che non hanno cicli attraverso la negazione — la complessità è PTIME, adatta al tempo reale.

22.6.3 Legge di Kirchhoff per Reti di Attenzione: una Interpretazione Elettrica

Un'interpretazione sorprendente del meccanismo di attenzione del transformer emerge dall'analogia con le reti elettriche: il flusso di informazione attraverso le teste di attenzione soddisfa una legge di conservazione analoga alla legge di Kirchhoff per le correnti. Sia $A \in \mathbb{R}^{n\times n}$ la matrice di attenzione normalizzata (stocastica per righe). L'output della testa è $Out_i = \Sigma_j A_{ij} V_j$, un media pesata dei valori. La matrice A rappresenta un operatore di diffusione: la sua iterazione A^t converge (per $t \to \infty$) alla misura stazionaria π dell'chain di Markov con matrice di transizione A, analoga alla distribuzione di probabilità stazionaria in un circuito resistivo.

Questa analogia ha implicazioni pratiche per la progettazione di architetture VLA robuste: layer di attenzione con spettro di A vicino a 1 (alta persistenza) corrispondono a circuiti con alta impedenza — trasferiscono poca informazione per step ma sono robusti al rumore. Layer con spettro lontano da 1 (bassa persistenza) corrispondono a amplificatori — trasferiscono molta informazione ma sono sensibili a perturbazioni. Il bilanciamento ottimale tra questi regimi — nell'architettura VLA — è un'area di ricerca aperta con connessioni alla teoria del segnale nei circuiti.

PARTE IX

Reinforcement Learning Avanzato per la Robotica Fisica

Capitolo 23 — Fondamenti del Reinforcement Learning per Sistemi Fisici

Il Reinforcement Learning (RL) per sistemi fisici — denominato Robot Reinforcement Learning o physical RL — affronta una sfida fondamentale che distingue la robotica dall'RL su giochi o simulazioni: il campionamento nel mondo reale è lento, costoso e potenzialmente pericoloso. Un robot che impara a camminare per trial-and-error senza vincoli di sicurezza rischia di danneggiare se stesso e l'ambiente. Questa parte sviluppa i fondamenti matematici del RL fisico, concentrandosi sulle tecniche di campionamento efficiente, sulla safe exploration e sull'integrazione con i controllori geometrici della Parte I.

23.1 Il Framework MDP per Sistemi Robotici Continui

Un sistema robotico umanoide è modellato come un Markov Decision Process (MDP) continuo $M = (S, A, P, R, \gamma)$ dove $S = TQ = \{(\eta, \dot{\eta})\}$ è lo spazio degli stati (coppia posizione-velocità), $A = \{\tau \in \mathbb{R}^n : |\tau_i| \leq \tau_{max}\}$ è lo spazio delle azioni (coppie articolari con limiti), $P: S \times A \to P(S)$ è il kernel di transizione, $R: S \times A \to \mathbb{R}$ è la funzione di reward e $\gamma \in (0,1)$ è il fattore di sconto. La funzione di reward per la locomozione bipede ha tipicamente la struttura:

$$R(s,a) = r_{vel} \cdot v_x + r_{energy} \cdot (-\|a\|^2) + r_{smooth} \cdot (-\|\dot{a}\|^2) + r_{alive} - r_{fall}$$

dove r_vel premia la velocità di avanzamento, r_energy penalizza il consumo energetico, r_smooth penalizza la non-smoothness delle azioni, r_alive è il reward di sopravvivenza e r_fall è la penalità per caduta. La progettazione della funzione di reward è un problema di reward engineering che richiede equilibrio tra gli obiettivi: un peso eccessivo su r_vel porta a politiche che corrono veloce ma cadono frequentemente; un peso eccessivo su r_energy porta a politiche che minimizzano il movimento.

La struttura continua dello spazio degli stati TQ impone un trattamento rigoroso della misurabilità. La distribuzione di transizione $P(\cdot|s,a)$ è una probabilità di Borel regolare su TQ: la stationarietà markoviana è

verificata per le equazioni di Eulero-Lagrange soggette a rumore di processo additivo w_t ~ N(0, Σ_w), il che produce un diffusion process con generatore infinitesimale L = Σ_i fi(s,a) ∂/∂si + (1/2) Σ_{ij} (Σ_w)_{ij} ∂²/∂si∂sj. La regolarità di questo generatore garantisce l'esistenza di soluzioni forti all'equazione di Hamilton-Jacobi-Bellman (HJB) associata.

La funzione valore ottima V*(s) = max_π E_π[Σt γt R(st, at) | s0 = s] soddisfa l'equazione HJB in forma viscosità: ϱ V*(s) = max_a [R(s,a) + (LV*)(s,a)], dove ϱ = −log γ è il tasso di sconto continuo. Per sistemi con dinamica differenziale della forma ṡ = f(s,a) + σ(s)wt, la soluzione V* esiste ed è unica nella classe delle funzioni lipschitziane continue, garantita dal teorema di Crandall-Lions per equazioni HJB degeneri. Il controllore ottimo in forma chiusa è quindi π*(s) = argmax_a [R(s,a) + ∂V*/∂s · f(s,a)], che per sistemi affini nel controllo produce la ben nota struttura bang-bang o lineare-quadratica a seconda della struttura della funzione reward.

La scelta della funzione reward merita un approfondimento algebrico. Il vettore di peso r = (r_vel, r_energy, r_smooth, r_alive, r_fall) ∈ ℝ5 definisce una funzione scalare sul prodotto TQ × A attraverso il mapping R(s,a) = r·φ(s,a), dove φ: TQ × A → ℝ5 è il vettore delle feature. La convessità del problema di reward engineering si riduce alla ricerca di r* = argmin_r E_π*_r[f_task], dove π*_r è la policy ottima per il reward pesato r e f_task è la metrica di task finale (velocità media, consumo energetico integrato, ecc.). Questo problema è in generale non convesso per la dipendenza di π*_r da r attraverso l'intera catena di Bellman, ma diventa trattabile nell'approssimazione lineare se le policy sono parametrizzate da una classe convessa.

Complessità campionaria e PAC-MDP

Il framework PAC-MDP (Probably Approximately Correct MDP, Kakade 2003) fornisce garanzie teoriche sul numero di passi necessari per apprendere una policy ε-ottima con probabilità almeno 1 − δ. Sia Nε,δ il numero di campioni sufficienti: per spazi di stato-azione continui con dinamica lipschitziana (costante L_f) e funzione reward lipschitziana (costante L_R), il risultato principale è Nε,δ = O((L_f L_R / ε²) · (dim(S) + dim(A)) · log(1/δ)), dove dim(S) e dim(A) sono le dimensioni di copertura

(covering numbers) degli spazi a risoluzione ε. Per un umanoide con n = 30 DOF si ha dim(TQ) ≈ 60, rendendo Nε,δ proibitivo per il campionamento diretto in ambiente reale — il che giustifica l'uso massiccio della simulazione parallela (§9.2, IsaacLab) e del model-based RL (§8.7, TD-MPC2).

Un risultato complementare riguarda il regime di bassa densità di campionamento (sparse sampling, Kearns-Singh 1999): con accesso a un simulatore della dinamica, è possibile costruire un albero decisionale di profondità H con branching factor K sufficiente a garantire un ritorno ε-ottimo in O((K/ε)^H) simulazioni, indipendentemente dalla dimensionalità di TQ. Questo bound, pur esponenziale in H, è polinomiale nel numero effettivo di stati visitabili nella traiettoria ottima — ed è la giustificazione teorica degli algoritmi MCTS applicati alla locomozione con world model (§8.7).

Proprietà della funzione reward in locomozione: analisi di sensibilità

Sia J(r) = E_{π*_r}[Σt γt v_x(st)] il ritorno medio sulla velocità di avanzamento quando la policy è ottima per il reward r. Il gradiente ∂J/∂r_energy misura quanto un incremento della penalità energetica degradi la velocità: per sistemi locomotori biologicamente plausibili, si ha tipicamente ∂J/∂r_energy < 0 e |∂J/∂r_vel| ≫ |∂J/∂r_smooth|, indicando che la velocità è il termine dominante nel tradeoff. La frontiera di Pareto nel piano (J_vel, J_energy) — parametrizzata da r_energy/r_vel — è una curva concava che corrisponde alla regione di efficienza locomotiva ottimale, analoga alla curva di offerta-domanda dell'economia del movimento di Todorov (2004).

Il termine r_smooth penalizza la norma quadratica delle variazioni temporali delle azioni: L_smooth(a,ȧ) = ‖ȧ‖² = ‖dτ/dt‖². Questo funzionale, quando integrato lungo la traiettoria, è equivalente alla minimizzazione dell'energia di jerk articolare, connettendo il design del reward alla meccanica del movimento biologico. Formalmente, minimizzare E[∫ ‖ȧ‖² dt] equivale a porre il prior sulle azioni come un processo di Wiener integrato (smooth Brownian motion), producendo traiettorie di coppia con derivata quadratica integrabile — che è la classe

di funzioni di Sobolev $H^1(0,T;\mathbb{R}^n)$, densa nello spazio delle traiettorie fisicamente realizzabili.

> Proposizione 23.1 (Continuità della policy ottima rispetto al reward). Sia $R(s,a) = r \cdot \varphi(s,a)$ con φ bounded e lipschitziano in s. Per ogni $\varepsilon > 0$ esiste $\delta > 0$ tale che $\|r_1 - r_2\| < \delta$ implica $\|V^*_{r_1} - V^*_{r_2}\|_\infty < \varepsilon$, dove V^*_r è la funzione valore ottima per il reward pesato r. La costante di continuità è $L_V \le \|\varphi\|_\infty / (1-\gamma)$.
>
> Dimostrazione. Dalla definizione di V^*_r e dall'equazione di Bellman: $|V^*_{r_1}(s) - V^*_{r_2}(s)| \le |\max_a [(r_1-r_2)\cdot\varphi(s,a) + \gamma\, E[V^*_{r_1}(s') - V^*_{r_2}(s')]]| \le \|r_1-r_2\|\cdot\|\varphi\|_\infty + \gamma\, \|V^*_{r_1} - V^*_{r_2}\|_\infty$. Raccogliendo il termine $\|V^*\|_\infty$ a sinistra: $\|V^*_{r_1} - V^*_{r_2}\|_\infty \le \|r_1-r_2\|\cdot\|\varphi\|_\infty / (1-\gamma)$. □

Questa proposizione ha implicazioni dirette per l'ingegneria del reward: piccole variazioni nei pesi r producono variazioni proporzionali nel valore atteso, con costante di amplificazione $1/(1-\gamma)$ che esplode per $\gamma \to 1$ (pianificazione a orizzonte infinito). Per $\gamma = 0.99$ (valore tipico in locomozione con frequenza di controllo 100 Hz), si ha $1/(1-\gamma) = 100$, il che implica che una variazione dell'1% nei pesi può produrre variazioni del 100% nel valore a lungo termine — giustificando la necessità di procedure sistematiche di reward tuning come EUREKA (§25.1) e RLHF (§5.4).

23.2 Policy Gradient per Sistemi Continui: TRPO e PPO

Gli algoritmi di policy gradient per spazi di azione continui ottimizzano la policy parametrica $\pi_\theta: S \to P(A)$ massimizzando il ritorno atteso $J(\theta) = E_\tau[\Sigma_t\, \gamma^t\, R(s_t,a_t)]$ tramite aggiornamenti del gradiente. L'estimatore del gradiente di policy è:

$$\nabla_\theta J(\theta) = E_\tau[\Sigma_t\, \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot A^\pi(s_t,a_t)]$$

dove $A^\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$ è la funzione di advantage. Il problema principale dell'aggiornamento tramite gradiente euclideo in θ è l'instabilità: passi troppo grandi in θ possono degradare drasticamente la policy, mentre passi troppo piccoli convergono lentamente.

TRPO: vincolo di KL-divergenza

Trust Region Policy Optimization (TRPO, Schulman et al., 2015) risolve il problema di aggiornamento con un vincolo esplicito sulla KL-divergenza dalla policy vecchia:

$$\theta_{k+1} = \arg\max_\theta L(\theta), \text{ s.t. } E_s[D_{KL}(\pi_{\theta_{old}}(\cdot|s) \,\|\, \pi_\theta(\cdot|s))] \le \delta$$

dove $L(\theta) = E[\pi_\theta/\pi_{\theta_{old}} \cdot A_{\pi_{old}}]$ è il surrogate objective (importanza pesata dell'advantage) e $\delta \approx 0.01$ è il radius della trust region. TRPO risolve questo problema constrained tramite l'approssimazione al secondo ordine e il metodo del gradiente naturale (§6.4), producendo aggiornamenti che rispettano la geometria dello spazio delle policy.

PPO: clipping per semplicità implementativa

Proximal Policy Optimization (PPO, Schulman et al., 2017) approssima il vincolo KL di TRPO con una funzione di clipping più semplice da implementare:

$$L_{PPO}(\theta) = E[\min(r_t(\theta) A_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon) A_t)]$$

dove $r_t(\theta) = \pi_\theta(a_t|s_t) / \pi_{\theta_{old}}(a_t|s_t)$ è il likelihood ratio e $\varepsilon \approx 0.2$ è il clip range. Il clipping impedisce aggiornamenti che portano r_t fuori dall'intervallo $[1-\varepsilon, 1+\varepsilon]$, limitando implicitamente la KL-divergenza. PPO è diventato il metodo standard per il training RL di policy locomotorie su IsaacLab (§9.2) per la sua semplicità, robustezza e scalabilità a miliardi di parametri.

Teorema 23.1 (Bound di miglioramento della policy per PPO).

Per ogni aggiornamento PPO con clip range ε e advantage normalizzato, il degradamento della policy soddisfa: $J(\theta_{new}) \ge J(\theta_{old}) - (4\varepsilon\gamma\, r_{max}) / (1-\gamma)^2 \cdot D_{KL,max}$, dove $D_{KL,max}$ è la massima KL-divergenza tra le policy e $r_{max} = \max_{s,a} |R(s,a)|$.

Dimostrazione. Il bound segue dal lemma di performance di Kakade-Langford: $J(\theta) - J(\theta_{old}) = (1/(1-\gamma)) E_{s\sim d_\theta}[E_{a\sim\pi_\theta}[A_{old}(s,a)]]$. Il surrogate objective approssima questo termine con importance sampling, e il clipping limita il peso massimo a $1+\varepsilon$. Il termine di discrepanza tra il surrogate e il vero miglioramento è bounded dalla divergenza KL tramite la disuguaglianza di Pinsker. □

Capitolo 24 — Curriculum Learning e Reward Shaping per la Locomozione

La locomozione bipede è un task di reinforcement learning particolarmente difficile per la sparsità del reward: il segnale di reward significativo (velocità di avanzamento) è ricevuto solo quando il robot cammina in modo coordinato, un comportamento che è estremamente raro da osservare con una policy casuale. Il curriculum learning e il reward shaping sono due tecniche complementari per guidare l'esplorazione verso comportamenti utili.

24.1 Curriculum Learning: Progressione della Difficoltà

Il curriculum learning (Bengio et al., 2009) addestra il sistema su una sequenza di task di difficoltà crescente, sfruttando il fatto che le soluzioni dei task più semplici forniscono una buona inizializzazione per i task più difficili. Per la locomozione bipede, un curriculum tipico è:

Livello 1: il robot è inizializzato già in posizione bipede eretta su terreno piano, con il reward che premia il mantenimento dell'equilibrio (r_alive) e piccoli spostamenti. Durata: 10^7 passi.

Livello 2: si aggiunge il reward per la velocità di avanzamento a bassa velocità (v_target = 0.3 m/s). Il terreno rimane piano. Durata: 10^7 passi.

Livello 3: si aumenta la velocità target (v_target = 0.8 m/s) e si introduce terreno con piccole irregolarità (range ±2 cm). Durata: 2×10^7 passi.

Livello 4: terreno con ostacoli e scale, velocità target variabile. Durata: 5×10^7 passi.

La progressione del curriculum è governata da una funzione di scheduling $\pi\_curr: \mathbb{N} \to [0,1]$ che mappa il numero di passi di training al livello di difficoltà. Lo scheduling ottimale è quello che mantiene la performance del robot appena al di sopra della soglia di successo (70–80%) — abbastanza difficile da stimolare l'apprendimento, abbastanza facile da evitare la divergenza.

Automatic Curriculum Learning: fondamenti teorici e connessione con il regret

Il framework PAIRED (Dennis et al. 2020) introduce il concetto di regret come metrica per la generazione automatica del curriculum. Il regret dell'agente π rispetto a una policy di riferimento π_ref nell'ambiente E è definito come $\text{Regret}(\pi, \pi\_ref, E) = V(\pi\_ref, E) - V(\pi, E)$, dove $V(\pi, E)$ è il valore atteso dell'agente π nell'ambiente E. L'agente antagonista genera ambienti $E^* = \text{argmax}\_E \text{Regret}(\pi, \pi\_ref, E)$ soggetto al vincolo $V(\pi\_ref, E) \geq V\_min$: il vincolo garantisce che l'ambiente sia risolubile dalla policy di riferimento (quindi non impossibile), mentre la massimizzazione del regret garantisce che l'ambiente sia sfidante per la policy corrente π.

La connessione con la teoria dell'apprendimento è profonda: il regret $\text{Regret}(\pi, \pi\_ref, E)$ misura quanto la policy π sia subottima nell'ambiente E rispetto al comportamento di riferimento. Massimizzare il regret equivale quindi a trovare gli ambienti in cui la policy è più lontana dall'ottimalità — ovvero gli ambienti di maggiore potenziale di apprendimento. Questa intuizione è formalizzata nel framework della Maximin Curriculum Theory (Portelas et al. 2019): il curriculum ottimale è quello che massimizza il tasso di riduzione del regret atteso $dG/dt = E\_E[d\text{Regret}(\pi\_t, \pi^*, E)/dt]$, che per politiche che convergono all'ottimo è massimizzato dagli ambienti per cui $\partial\text{Regret}/\partial\theta\_\pi$ ha norma massima — i.e., gli ambienti più informativi per l'aggiornamento dei parametri.

> Teorema 24.1 (Convergenza del curriculum basato su regret). Sotto le ipotesi che la classe di ambienti E sia compatta, che il valore $V(\pi, E)$ sia differenziabile in θ (parametri di π) e in φ (parametri dell'ambiente), e che l'agente antagonista risolva esattamente il

problema di massimizzazione del regret, l'agente principale converge a una policy ε-ottimale su tutta la distribuzione p(E) in un numero di passi $T = O(1/\varepsilon^2 \cdot C\_class)$, dove C_class è la complessità della classe di policy (ad esempio la dimensione VC).

Dimostrazione. Il bound segue dal framework PAC-learning applicato alla sequenza di ambienti generati dal curriculum. Ad ogni iterazione, il curriculum genera ambienti con alta informazione (alto regret), il che equivale a campionare nella regione dello spazio degli ambienti dove il gradiente della policy è più ampio — minimizzando la varianza dell'estimatore del gradiente nel senso della reduced-variance optimization. Il tasso di convergenza $O(1/\varepsilon^2)$ segue dal gradiente stocastico con varianza ridotta. □

Il legame con l'ottimizzazione di traiettoria del Capitolo 3 emerge nella struttura del problema dell'agente antagonista: la generazione di ambienti sfidanti per la locomozione bipede corrisponde alla selezione di profili di terreno (pendenze, irregolarità, ostacoli) che massimizzano il regret. Questo è formalmente equivalente al problema di sintesi degli stimoli ottimali (optimal experiment design) nella system identification (§21.1): il curriculum ottimale per la locomozione sceglie i terreni che massimizzano il Fisher information sulla policy, producendo aggiornamenti parametrici di massima efficienza — e la connessione con le traiettorie PE (Persistently Exciting, §21.1) è precisa: ambienti che massimizzano il regret sono esattamente gli ambienti che massimizzano l'informazione di Fisher sulla policy corrente.

Automatic Curriculum Learning (ACL)

L'Automatic Curriculum Learning (ACL) sostituisce il curriculum manuale con un algoritmo che adatta automaticamente la difficoltà basandosi sulla performance corrente. L'idea di PAIRED (Dennis et al., 2020) è usare un agente antagonista che genera ambienti difficili per il robot, vincolato a produrre ambienti che il robot riesce a risolvere con probabilità θ_min (evitando ambienti impossibili):

$$\max_{\varphi} [J(\pi_{\theta}, E_{\varphi}) - J(\pi_{ref}, E_{\varphi})], \text{ s.t. } J(\pi_{ref}, E_{\varphi}) \geq \theta_{min}$$

dove E_φ è l'ambiente generato con parametri φ, π_θ è la policy del robot e π_ref è una policy di riferimento più semplice. Questo approccio è formalizzazione del regret maximization: l'agente antagonista massimizza il regret (la differenza tra la performance del robot e quella di riferimento), il che produce ambienti al limite della capacità del robot — il punto di massimo apprendimento.

24.2 Reward Shaping: Potenziali e Invarianza

Il reward shaping aggiunge un termine ausiliario F(s, s') alla funzione di reward R(s,a) per guidare l'esplorazione verso stati desiderabili. Il problema del reward shaping è che modificare il reward può cambiare la policy ottimale: un robot addestrato con il reward sbagliato potrebbe imparare a massimizzare il reward ausiliario invece del reward originale. Il teorema di Ng-Harada-Russell (1999) caratterizza le trasformazioni di reward che preservano la policy ottimale.

Teorema 24.1 (Reward Shaping Invariance, Ng et al. 1999).

Dato un MDP M con reward R, la policy ottimale π^* di M è invariata se si aggiunge il reward shaping $F(s, a, s') = \gamma\,\Phi(s') - \Phi(s)$ per qualsiasi funzione di potenziale $\Phi: S \to \mathbb{R}$. Inoltre, questa è l'unica classe di reward shaping che garantisce l'invarianza per tutti gli MDP.

Dimostrazione. La funzione di valore del reward modificato è Q_F(s,a) = Q(s,a) + γ Φ(s') − Φ(s). Per ogni politica π, il valore relativo tra coppie stato-azione è $Q\_F(s,a_1) - Q\_F(s,a_2) = Q(s,a_1) - Q(s,a_2)$, indipendentemente da Φ. Quindi argmax_a Q_F(s,a) = argmax_a Q(s,a) per ogni s. La necessità segue dal fatto che qualsiasi reward shaping che non sia di questa forma produce MDP per cui le policy ottimali differiscono. □

Per la locomozione bipede, un potenziale di Φ naturale è la distanza al goal: $\Phi(s) = -\|p\_robot - p\_goal\|$. Questo produce il reward shaping F(s,s') = γ $\|p\_robot(s) - p\_goal\| - \|p\_robot(s') - p\_goal\|$, che ricompensa ogni avanzamento verso il goal e penalizza ogni allontanamento, guidando l'esplorazione nella direzione giusta senza modificare la policy ottimale.

Reward shaping adattivo e connessione con il trasporto ottimale

Il reward shaping tramite potenziale $\Phi(s)$ è equivalente, dal punto di vista della struttura delle traiettorie ottimali, alla modifica della misura di trasporto nel problema di pianificazione stocastica. Sia μ_0 la distribuzione iniziale degli stati e μ_T la distribuzione target (goal). Il problema di RL è di trovare la policy π^* che trasporti μ_0 a μ_T minimizzando il costo cumulativo: questo è formalmente un problema di trasporto ottimale stocastico in cui il costo $c(s,s') = R(s,a) - \log \pi(a|s)$. Il reward shaping con potenziale Φ aggiunge il termine $\gamma\Phi(s') - \Phi(s)$ al costo, il che corrisponde a una trasformazione di gauge del potenziale di trasporto: nel framework del trasporto ottimale, questo è equivalente a modificare le condizioni al contorno del problema senza cambiare le traiettorie ottimali. Il Teorema di Ng-Harada-Russell è quindi un caso speciale dell'invarianza di gauge del trasporto ottimale.

Una classe particolarmente efficace di reward shaping per la locomozione è quella dei potenziali di distanza geodetica: $\Phi(s) = -d\_geo(s, s\_goal)$, dove d_geo è la distanza geodetica nella varietà dello spazio degli stati rispetto alla metrica di Riemannian indotta dall'energia cinetica del robot. Questo potenziale premia il moto nella direzione di massima pendenza della distanza geodetica — ovvero nella direzione delle geodetica che connette lo stato corrente al goal — senza richiedere la conoscenza esplicita della traiettoria ottima. La connessione con il formalismo di Lie (Capitolo 15) emerge dal fatto che la distanza geodetica nella varietà delle configurazioni Q è quella indotta dalla metrica di Riemannian $M(\eta)$ — la stessa metrica che definisce i simboli di Christoffel (§1.4) e le geodetiche del moto libero (§1.4, connessione con le equazioni del moto libero $M(\eta)\ddot{\eta} + C(\eta,\dot{\eta})\dot{\eta} = 0$ come equazione delle geodetiche).

> Proposizione 24.2 (Ottimalità del potenziale geodetico per reward shaping). Sia $d\_geo(s, s\_goal)$ la distanza geodetica nella varietà $(Q, M(\eta))$ e $\Phi(s) = -d\_geo(s, s\_goal)$ il potenziale geodetico. Il reward shaping $F(s,s') = \gamma\ \Phi(s') - \Phi(s)$ è il potenziale di shaping che massimizza il tasso di avanzamento verso il goal $E[d/dt\ \Phi(s(t))]$ tra tutti i potenziali lipschitziani con costante $L \le L\_\Phi = \|\nabla\Phi\|\_\infty$.

Dimostrazione. Il tasso di avanzamento verso il goal è $d/dt\ \Phi(s) = \nabla\Phi(s) \cdot f(s,a)$, dove f è la dinamica del sistema. Il potenziale che massimizza questo tasso puntualmente è quello con gradiente allineato alla dinamica ottima verso il goal: $\nabla\Phi(s) = f_{opt}(s) / \|f_{opt}(s)\| \cdot L_\Phi$. Per sistemi differenziali con dinamica affine $f(s,a) = f_0(s) + g(s)u$, il gradiente della distanza geodetica $-\nabla d_{geo}(s, s_{goal})$ è esattamente la direzione di moto ottimo sulla geodetica che connette s a s_{goal}, e ha norma 1 per la metrica M — che coincide con $L_\Phi = 1$ per potenziali con gradiente unitario. □

Il reward shaping adattivo estende il framework statico introducendo un potenziale $\Phi_t(s)$ che si aggiorna durante il training in funzione della distribuzione di stati visitati. Un approccio è il Potential-Based Reward Shaping con hindsight (PBRS-H): $\Phi_t(s) = E_{\tau \sim \pi_t}[\Sigma_{t' \geq t} \gamma^{t'-t} R(s_{t'}, a_{t'}) \mid s_t = s] - E_{\tau \sim \pi_t}[\Sigma_{t' \geq 0} \gamma^{t'} R(s_{t'}, a_{t'}) \mid s_0 = s_{init}]$, ovvero il potenziale è la funzione di advantage $A_{\pi t}(s)$. L'invarianza del Teorema di Ng garantisce che questo shaping non modifica la policy ottimale, ma la funzione di reward modificata $R + F$ è più densa dell'originale R (riceve feedback positivo in ogni stato dove il ritorno futuro è superiore alla media), riducendo la varianza del policy gradient di un fattore proporzionale alla riduzione della varianza del reward (teorema di baseline subtraction per policy gradient).

Capitolo 25 — Reinforcement Learning con Large Language Models come Reward Models

Un approccio emergente nel 2024-2025 è l'utilizzo di Large Language Models (LLM) come funzioni di reward per il RL robotico: invece di specificare manualmente la funzione di reward, si descrive il task in linguaggio naturale e si usa il LLM per valutare le traiettorie del robot. Questo approccio — denominato Language-Guided Reward Learning (LGRL) — riduce significativamente l'ingegneria del reward e consente di specificare task complessi e sfumati in linguaggio naturale.

25.1 LLM come Oracle di Reward

In EUREKA (Ma et al., 2023), il sistema usa GPT-4 per generare automaticamente codice Python che implementa la funzione di reward, basandosi sulla descrizione del task e sul codice del simulatore. Il ciclo è:

Passo 1: l'utente descrive il task in linguaggio naturale ("il robot deve aprire il cassetto senza danneggiare gli oggetti intorno").

Passo 2: GPT-4 genera una funzione di reward r(s,a) come codice Python, analizzando l'API del simulatore e la descrizione del task.

Passo 3: la funzione di reward viene testata nel simulatore per K episodi. Le curve di reward, le metriche di task e i video vengono restituiti a GPT-4 come feedback.

Passo 4: GPT-4 analizza il feedback e genera una versione migliorata della funzione di reward. Il ciclo si ripete per T iterazioni.

Il risultato riportato in Ma et al. (2023) è che EUREKA supera le funzioni di reward progettate manualmente da esperti nel 83% dei 29 task testati, inclusi task come pen spinning, ball balancing e manipulation in dexterous hands, per i quali il design manuale del reward era considerato particolarmente difficile.

La generazione automatica di reward tramite LLM si inserisce nel più ampio problema della reward specification — la formalizzazione precisa

delle preferenze umane in un linguaggio computazionalmente trattabile. Il problema è noto in letteratura come alignment problem nella sua componente di reward design: qualsiasi discrepanza tra la funzione di reward specificata e le reali preferenze dell'utente può portare a comportamenti indesiderati anche con ottimizzazione perfetta della policy (Goodhart's Law: when a measure becomes a target, it ceases to be a good measure). Il contributo tecnico di EUREKA è di trasformare il problema di reward specification in un problema di code generation, sfruttando la capacità degli LLM di ragionare su strutture simboliche e di tradurre descrizioni naturali in vincoli computazionali.

L'architettura di EUREKA può essere formalizzata come un processo di ottimizzazione bayesiana su uno spazio di funzioni reward: sia F lo spazio delle funzioni r: S × A → $\mathbb{R}$ implementabili come codice Python, e sia J: F → $\mathbb{R}$ la metrica di task finale (ad esempio, la velocità media di avanzamento o il tasso di successo su un benchmark). Il processo LLM definisce una distribuzione prior P_LLM(r | task_description, sim_api) sullo spazio F, condizionata sulla descrizione del task e sull'API del simulatore. L'obiettivo è trovare r* = argmax_{r ∈ F} J(r) con il minor numero possibile di valutazioni della funzione di merito J — ciascuna valutazione richiede il training di una policy tramite RL, costoso in termini computazionali.

Il ciclo di feedback come ottimizzazione bayesiana

Il loop EUREKA può essere visto come un algoritmo di ottimizzazione bayesiana (BO) in cui il surrogate model è implementato implicitamente dall'LLM. Ad ogni iterazione t, l'LLM osserva la storia delle funzioni reward valutate $\{r_1,...,r_◎\}$ e le corrispondenti metriche $\{J(r_1),...,J(r_◎)\}$, insieme ai failure mode osservati (ad esempio, il robot che sfrutta reward shaping in modi non previsti), e genera la prossima proposta r_{t+1} mediante il processo di generazione autoregressiva. Questo schema corrisponde alla regola di acquisition di Thompson Sampling nel framework BO: l'LLM campiona implicitamente una funzione di acquisizione basata sull'incertezza e il potenziale di miglioramento stimato dalla sua distribuzione prior aggiornata.

La convergenza del ciclo EUREKA può essere analizzata nel framework del regret cumulativo di Bayesian Optimization. Sia r_T* il massimizzatore trovato dopo T iterazioni e r* il massimizzatore globale. Il regret cumulativo è R_T = Σ_{t=1}^T [J(r*) − J(r_t)]. Per surrogate model con prior gaussiano e funzione di acquisizione UCB (Upper Confidence Bound), il regret cumulativo soddisfa R_T = O(√(T γ_T)) dove γ_T è la maximum information gain dopo T iterazioni. Per funzioni di merito lipschitziane su spazi di reward con struttura di kernel a bassa complessità, γ_T = O(log T), producendo R_T = O(√(T log T)) — sublineare in T, il che garantisce la convergenza. Nel caso di EUREKA, l'LLM implementa un surrogate model non-parametrico con struttura di prior ricca, che in pratica permette di convergere in 3–10 iterazioni su task standard.

> Teorema 25.1 (Convergenza di EUREKA con feedback strutturato). Sia F la classe delle funzioni reward implementabili in Python con complessità di Kolmogorov al più K_max bits, e sia J lipschitziana in norma L∞ su F con costante L_J. Dopo T iterazioni del ciclo EUREKA con feedback contenente la traiettoria di training, le metriche di task decomposte per fase e i failure mode identificati, l'errore di approssimazione soddisfa E[J(r*) − J(r_T)] = O(L_J · 2^{−αT}) per una costante $\alpha > 0$ che dipende dalla qualità del prior dell'LLM sulla classe F.
>
> Dimostrazione (sketch). La dimostrazione richiede di modellare l'LLM come un campionatore dalla distribuzione posterior P_LLM(r | D_t), dove D_t è la storia delle osservazioni. Sotto l'ipotesi che l'LLM abbia prior consistente con F (ovvero che P_LLM(r* | ·) > 0 per ogni r* ∈ F), la massa di probabilità attorno al massimizzatore cresce ad ogni iterazione con tasso almeno esponenziale nella quantità di informazione fornita dal feedback strutturato. L'esponente α è limitato inferiormente dalla quantità di informazione mutua I(r_t; J(r_t) | D_{t-1}) per iterazione. □

Il feedback strutturato è il componente critico che separa EUREKA dai semplici generatori di codice. Invece di restituire soltanto il valore scalare J(r_t), il feedback include: (1) le curve di apprendimento delle policy intermedie, che rivelano se il reward è troppo sparso o troppo denso; (2) la

decomposizione del reward per componente fisica, che identifica quali termini del reward vengono ottimizzati e quali ignorati; (3) i comportamenti anomali (reward hacking) osservati nelle politiche addestrate, descritti in linguaggio naturale tramite un pipeline di analisi automatica. Questa ricchezza informativa permette all'LLM di identificare la causa del fallimento e di proporre correzioni mirate, a differenza dell'ottimizzazione gradient-free classica che osserva solo il valore scalare.

Connessione con la teoria dell'inverse reinforcement learning

EUREKA può essere interpretato come una forma di Inverse Reinforcement Learning (IRL) tramite LLM: invece di inferire il reward dalle dimostrazioni dell'esperto (come nell'IRL classico di Ng-Russell), il reward viene inferito dalla descrizione linguistica dell'intenzione dell'utente. La connessione formale si stabilisce tramite il principio di massima entropia (MaxEnt IRL, Ziebart et al. 2008): il reward ottimo r* è quello che massimizza l'entropia della distribuzione di policy soggetta al vincolo che le traiettorie dimostrate abbiano valore atteso corrispondente all'intenzione dichiarata. L'LLM implementa implicitamente questo principio attraverso il suo prior linguistico: le descrizioni naturali di comportamenti desiderabili corrispondono a vincoli sulla distribuzione di policy che l'LLM ha appreso durante il pre-addestramento su testi di fisica, ingegneria e biomeccanica.

25.2 VLM come Valutatori Visivi di Traiettorie

Un'alternativa ai reward basati su LLM è l'uso di Vision-Language Models (VLM) come valutatori visivi: dato un video della traiettoria del robot, il VLM produce un punteggio di successo task. Il vantaggio è che il VLM può valutare proprietà qualitative difficili da esprimere con formule matematiche: "il movimento è fluido e naturale?", "il robot evita di urtare gli oggetti?", "la postura è ergonomica?".

Il sistema RL-VLM-F (Rocamonde et al., 2023) usa CLIP per calcolare la similarità visivo-linguistica tra lo stato corrente del robot e l'immagine di goal:

$$R_{CLIP}(s_◎, g) = \cos(f_V(I(s_◎)), f_L(g)) / \tau$$

dove I(s◎) è il rendering del robot nello stato s◎, f_V e f_L sono gli encoder visivo e linguistico di CLIP, g è la descrizione testuale del goal e τ è la temperatura. Questo reward CLIP può essere usato come shaping reward in aggiunta al reward di task, guidando il robot verso stati visivamente simili alla descrizione del goal anche quando il reward sparso di task non è disponibile.

25.3 Analisi Teorica: Alignment tra Reward LLM e Reward Umano

Un problema fondamentale del LGRL è l'alignment: il reward generato dall'LLM può non corrispondere esattamente all'intenzione umana, producendo comportamenti che massimizzano il reward LLM ma non soddisfano l'utente (reward hacking). Il framework del Reward Model Uncertainty (RMU) quantifica questo rischio tramite la varianza del reward model:

$$R_{safe}(s,a) = E_{LLM}[r(s,a)] - \beta \sqrt{Var_{LLM}[r(s,a)]}$$

dove il termine di penalizzazione $\beta \sqrt{Var_{LLM}[r]}$ riduce il reward in regioni dove il LLM è incerto, spingendo la policy verso regioni familiari dove il LLM ha confidenza. Questo è analogo all'approccio Pessimism in the Face of Uncertainty (PUFU) in offline RL, dove l'incertezza del modello è usata per penalizzare le azioni rischiose.

Proposizione 25.1 (Bound di reward hacking con RMU).

Sia π^* la policy ottimale per il reward umano R_{human} e π_{LLM} la policy che massimizza il reward LLM R_{LLM}. Se $\|R_{human} - E_{LLM}[r]\|_\infty \le \varepsilon_{align}$ e la policy usa il reward conservativo R_{safe} con $\beta = \sqrt{(2 \log(1/\delta) / K)}$, allora con probabilità $\ge 1 - \delta$: $J_{human}(\pi_{LLM}) \ge J_{human}(\pi^*) - 2\varepsilon_{align} H / (1 - \gamma)$.

Dimostrazione. Il bound segue dalla decomposizione del performance gap in due termini: il misalignment tra reward LLM e reward umano (bounded da ε_{align} per costruzione), e la suboptimalità della policy rispetto al reward LLM (bounded da zero per costruzione). Il termine β nella penalizzazione di varianza garantisce che la policy eviti le regioni di alta incertezza dove ε_{align} potrebbe essere maggiore della media, con probabilità $1 - \delta$ tramite la disuguaglianza di concentrazione di Bernstein. □

Capitolo 26 — Multi-Agent RL e Coordinamento di Sistemi Robotici

La maggioranza dei risultati della monografia riguarda un singolo robot umanoide. Scenari applicativi reali — magazzini automatizzati, assistenza collaborativa, costruzione — richiedono la coordinazione di più robot che operano nello stesso ambiente fisico. Il Multi-Agent Reinforcement Learning (MARL) estende il framework MDP a sistemi di N agenti con obiettivi individuali (competitivi) o collettivi (cooperativi) o misti.

26.1 Stochastic Games e la Generalizzazione dell'MDP

Un Stochastic Game (SG, Shapley 1953) per N agenti è la tupla $\langle S, A^1,...,A^N, P, R^1,...,R^N, \gamma \rangle$ dove A^i è lo spazio di azione dell'agente i, $P: S \times A^1 \times ... \times A^N \to P(S)$ è la transizione condivisa e $R^i: S \times A^1 \times ... \times A^N \to \mathbb{R}$ è il reward individuale. La scelta del concetto di soluzione distingue tra:

Giochi cooperativi: gli agenti massimizzano il reward congiunto $J(\pi) = E[\Sigma_i \Sigma_{\odot} \gamma^t R^i]$. La soluzione ottimale è il vettore di policy $(\pi^{1*},...,\pi^{N*})$ che massimizza J. Con full observability e comunicazione libera, il problema si riduce a un MDP centralizzato di dimensione $|A^1| \times ... \times |A^N|$.

Giochi competitivi (zero-sum): $\Sigma_i R^i = 0$. La soluzione è l'equilibrio di Nash minimax: nessun agente può migliorare il proprio reward con un cambio unilaterale di strategia. Per giochi a due agenti, l'equilibrio di Nash minimax è unico (Teorema di von Neumann-Morgenstern) ed è calcolabile in PTIME tramite programmazione lineare.

La teoria degli Stochastic Games (SG) generalizza quella degli MDP in modo non banale: la presenza di molteplici agenti con possibili obiettivi conflittuali introduce discontinuità strutturali nell'insieme delle soluzioni

che non hanno analogo nel caso mono-agente. Il teorema di esistenza dell'equilibrio di Nash per SG a somma zero a orizzonte infinito con sconto è dovuto a Shapley (1953) e costituisce la fondazione teorica di questo framework. La sua dimostrazione usa un argomento di punto fisso di Brouwer applicato alla value iteration nei giochi: la mappa di Bellman per giochi a due agenti $T(V)(s) = \max_{\pi^1} \min_{\pi^2} [R(s, a^1, a^2) + \gamma \Sigma_{s'} P(s'|s,a^1,a^2) V(s')]$ è una contrazione in norma L^∞ con fattore $\gamma < 1$, e il suo unico punto fisso è il valore dell'equilibrio di Nash minimax V^*.

Per giochi cooperativi a N agenti, la funzione valore collettiva $W^*(s) = \max_{\pi^1,...,\pi^N} E[\Sigma t\, \gamma t\, \Sigma i\, R^i(st, a^1t,...,a^Nt) \mid s0 = s]$ soddisfa un'equazione di Bellman multi-agente che si riduce, nella fattorizzazione completa degli spazi di azione, a un problema di ottimizzazione combinatoria NP-hard (joint action space di dimensione $\Pi i\, |A^i|$). La centralizzazione completa elimina questa difficoltà riducendo il SG cooperativo a un MDP con spazio di azione aumentato, ma non è scalabile: per N = 10 robot e $|A^i| = 100$ azioni discrete ciascuno, lo spazio congiunto ha $100^{10} = 10^{20}$ elementi, rendendo la ricerca diretta computazionalmente proibitiva. Il paradigma CTDE (§26.2) risolve questo problema separando la fase di training (centralizzata, con informazione globale) da quella di esecuzione (decentralizzata, con solo osservazioni locali).

La teoria dei potenziali e la convergenza dei giochi cooperativi

Una classe speciale di SG cooperativi che ammette una teoria di convergenza particolarmente elegante è quella dei Potential Games (Monderer-Shapley, 1996). Un gioco è un potential game se esiste una funzione di potenziale $\Phi: S \times A^1 \times ... \times A^N \to \mathbb{R}$ tale che $R^i(s, a^i, a_{-i}) - R^i(s, \bar{a}^i, a_{-i}) = \Phi(s, a^i, a_{-i}) - \Phi(s, \bar{a}^i, a_{-i})$ per ogni coppia di azioni $(a^i, \bar{a}^i)$ del giocatore i e ogni profilo degli altri a_{-i}. La condizione di potential game implica che gli equilibri di Nash coincidano con i massimizzatori del potenziale Φ, rendendo il problema di trovare equilibri equivalente a un problema di ottimizzazione scalare.

Per sistemi robotici cooperativi, la condizione di potential game è verificata in scenari di coverage control: N robot devono coprire un insieme di punti target minimizzando il costo collettivo di copertura $C(p^1,...,p^N) = \Sigma_x \min_i \|x - p^i\|^2$. In questo caso, il potenziale è $\Phi = -C$, e le

policy di gradient ascent convergono all'unico equilibrio di Nash (la partizione di Voronoi ottima per la densità di punti target). Per estensione, i task robotici che hanno struttura di coverage — ad esempio, il monitoraggio distribuito di una fabbrica o la coordinazione di bracci multipli su un piano di lavoro condiviso — possono essere analizzati nel framework dei potential games, con garanzie di convergenza.

> Teorema 26.1 (Convergenza negli Stochastic Potential Games). Sia il gioco multi-agente un potential game con funzione di potenziale Φ strettamente concava in $(a^1,...,a^N)$ per ogni s. Allora: (i) esiste un unico equilibrio di Nash puro $\pi^* = (\pi^{*1},...,\pi^{*N})$; (ii) il processo di best-response dynamics (ogni agente aggiorna la propria policy massimizzando il proprio reward dati i profili degli altri) converge a π^* da qualsiasi inizializzazione; (iii) nel framework stocastico con sconto γ, il valore di potenziale $V_\Phi(s) = \max_{\pi} E_\pi[\Sigma t\ \gamma t\ \Phi(st, a^t)]$ soddisfa $V_\Phi = V^*$ dove V^* è il valore dell'equilibrio.
>
> Dimostrazione. (i) L'unicità segue dalla stretta concavità di Φ: il massimizzatore è unico, e per la proprietà del potential game coincide con l'unico equilibrio di Nash. (ii) Best-response dynamics incremente monotonamente Φ ad ogni step (ogni agente sceglie l'azione che massimizza il suo reward, equivalentemente incrementando Φ per la definizione di potential game). Poiché Φ è bounded superiormente (giochi con reward bounded), la monotona incrementale converge. (iii) La funzione valore del potential game soddisfa $V_\Phi(s) = \max_\pi E[\Sigma\ \gamma t\ \Phi(st,a^t)] = V^*(s)$ per definizione, dove V^* è costruita tramite la value iteration standard. □

Il caso dei giochi a somma zero per robotica emerge nelle applicazioni di adversarial testing e safety verification: un agente avversario (il disturbo) cerca di massimizzare il rischio per il robot, mentre il robot cerca di minimizzarlo. Questo schema è formalmente identico al controllo $H\infty$ (§17.1) e alla reachability analysis Hamilton-Jacobi-Isaacs: il valore del gioco $V^*(s)$ coincide con la viscosity solution dell'equazione HJI, e la policy minimax del robot coincide con il controllore safe ottimale. Il framework degli Stochastic Games fornisce quindi il collegamento matematico preciso tra control theory (sicurezza) e game theory

(comportamento multi-agente), unificando i Capitoli 17 e 26 di questa monografia.

Un caso particolare importante per la robotica collaborativa è il gioco di Stackelberg, in cui un agente leader annuncia la propria strategia prima che i follower reagiscano. Per sistemi robotici gerarchici — ad esempio, un robot pianificatore (leader) e N robot esecutori (follower) — l'equilibrio di Stackelberg è più efficiente dell'equilibrio di Nash simultaneo: il leader internalizza l'effetto della propria strategia sulle risposte dei follower, producendo un coordinamento implicito senza necessità di comunicazione esplicita. L'equilibrio di Stackelberg è caratterizzato dalla condizione di backward induction: il leader sceglie $\pi^{1*} = \text{argmax}_{\{\pi^1\}} J^1(\pi^1, \pi^{2*}(\pi^1))$ dove $\pi^{2*}(\pi^1)$ è la best-response del follower alla strategia del leader.

26.2 CTDE: Centralised Training con Decentralised Execution

Il paradigma Centralised Training, Decentralised Execution (CTDE) è il framework più adottato nel MARL cooperativo: durante il training si usa informazione globale (stati di tutti gli agenti), durante l'esecuzione ogni agente osserva solo la propria osservazione locale. MAPPO (Multi-Agent PPO) e MADDPG (Multi-Agent DDPG) sono le varianti CTDE dei corrispondenti algoritmi single-agent.

In MAPPO, il critic centralizzato $V_c(s^1,...,s^N)$ stima il valore congiunto dalla prospettiva dello stato globale, mentre la policy π^i_θ di ogni agente osserva solo o^i. La loss è:

$$L_{MAPPO}(\theta) = \Sigma_i \, E[\min(r^i(\theta) A^i, \text{clip}(r^i(\theta), 1-\varepsilon, 1+\varepsilon) A^i)]$$

dove $A^i = Q^i_c(s, a_i, a_{-i}) - V_c(s)$ è l'advantage centralizzato che usa le azioni di tutti gli agenti a_{-i}. Questa stima dell'advantage riduce la varianza rispetto all'advantage calcolato solo sull'osservazione locale, migliorando la convergenza del training.

26.3 Coordinamento tramite Comunicazione: Graph Neural Networks

Per robot umanoidi in ambienti condivisi, la comunicazione esplicita tra agenti è possibile tramite reti wireless o protocolli robotici. Il coordinamento ottimale richiede la condivisione di informazioni rilevanti senza sovraccaricare la rete di comunicazione. Graph Neural Networks (GNN) sono uno strumento naturale per questo: ogni robot è un nodo del grafo, le comunicazioni sono gli archi, e la policy di comunicazione apprende quali informazioni trasmettere.

Il framework CommNet (Sukhbaatar et al., 2016) implementa la comunicazione come una media degli hidden states dei vicini:

$$h^{i}_{+1} = f(h^{i}, (1/|N(i)|) \Sigma_{\in N(i)} c^{i,j} h^{j})$$

dove N(i) è il vicinato del robot i, $c^{i,j}$ è il vettore di comunicazione da j ad i, e f è una rete ricorrente (GRU o LSTM). Le policy individualii sono poi derivate dagli hidden states aggiornati con comunicazione. La convergenza di CommNet per il coordinamento cooperativo è garantita per grafi connessi con regolarità sufficiente della funzione di valore congiunto.

L'architettura a grafo per la comunicazione multi-agente formalizza il problema di coordinamento come un processo di message-passing su un grafo G = (V, E) dove V = {1,...,N} è l'insieme degli agenti ed E ⊆ V×V è l'insieme dei canali di comunicazione disponibili. La GNN opera tramite L round di message-passing: al round l, l'hidden state $h^{i}_{(l+1)}$ = UPDATE($h^{i}_{(l)}$, AGGREGATE({$h^{j}_{(l)}$: j ∈ N(i)})), dove N(i) è il vicinato dell'agente i nel grafo. La funzione di AGGREGATE deve essere invariante per permutazione del vicinato (sum, mean, max sono le scelte standard); UPDATE è tipicamente una GRU o LSTM che integra il messaggio aggregato con lo stato nascosto locale.

La capacità espressiva delle GNN per il coordinamento multi-agente è caratterizzata dalla potenza del test di isomorfismo di grafi WL (Weisfeiler-Leman). Due grafi non isomorfi sono distinguibili da una GNN standard con AGGREGATE = sum se e solo se sono distinguibili dal WL test. Questo implica che strutture di coordinamento con simmetria di gruppo (ad esempio, N robot identici in formazione circolare) non sono distinguibili dalla GNN e producono policy identiche per tutti gli agenti — il che è desiderabile se si vuole una formazione simmetrica, ma può

essere limitante per task asimmetrici. L'aumento di espressività tramite random node features (GNN+RNF) o higher-order GNN (k-GNN) risolve questa limitazione a costo di maggiore complessità computazionale.

Message-passing ottimale e its connessione con belief propagation

Il message-passing nelle GNN ha una connessione profonda con il belief propagation (BP) dell'inferenza su grafi probabilistici. Nel framework BP, il messaggio dall'agente j all'agente i al round l è $m_{j\to i}^{(l)} = \Sigma_{x^j} \psi_{ij}(x^i, x^j) \Pi_{k\in N(j)\setminus\{i\}} m_{k\to j}^{(l-1)}$, dove ψ_{ij} è il fattore di compatibilità locale e x^i, x^j sono le variabili latenti degli agenti. Quando la GNN apprende le funzioni UPDATE e AGGREGATE, sta implicitamente apprendendo un sistema di messaggi ottimali per coordinare le decisioni degli agenti — ovvero i fattori di compatibilità ψ_{ij} che massimizzano il reward congiunto nel framework di un graphical model. Per grafi con struttura ad albero, il BP converge in esattamente l passi (dove l è la profondità dell'albero), garantendo la computazione esatta della distribuzione marginale di ogni agente. Per grafi con cicli, il loopy BP approssima le marginali con precisione che cresce con il numero di round, producendo GNN che migliorano monotonamente la qualità del coordinamento all'aumentare della profondità L.

> Teorema 26.2 (Universalità delle GNN di coordinamento). Sia G = (V,E) un grafo di comunicazione con N nodi. Una GNN con L layer, dimensione nascosta d sufficientemente grande e funzioni UPDATE/AGGREGATE apprendibili può approssimare con errore ε arbitrario qualsiasi policy di coordinamento π: $O^1\times...\times O^N \to A^1\times...\times A^N$ tale che $\pi(o_{\sigma(1)},...,o_{\sigma(N)}) = \sigma(\pi(o^1,...,o^N))$ per ogni permutazione σ del vicinato (policy equivariante al grafo), usando un numero di parametri $O(N\, d^2\, L)$.
>
> Dimostrazione. La dimostrazione procede in due passi. (1) Per $L \geq \mathrm{diam}(G)$ (diametro del grafo), ogni agente riceve informazioni su tutti gli altri agenti tramite il message-passing: $h^i_{(L)}$ dipende da tutti gli hidden states iniziali $\{h^j_{(0)}\}_{j\in V}$. Quindi la GNN è globalmente informativa. (2) Per d sufficientemente grande e funzioni UPDATE/AGGREGATE universalmente approssimanti (MLP), la GNN con L layer può approssimare qualsiasi funzione

continua sui vettori di hidden state — che include qualsiasi policy di coordinamento continua equivariante al grafo. Il numero di parametri $O(N\, d^2\, L)$ segue dalla struttura a peso condiviso: i pesi delle funzioni UPDATE e AGGREGATE sono condivisi tra tutti gli agenti (equivarianza). □

La scelta dell'architettura di comunicazione (struttura del grafo E) ha impatto diretto sulla qualità del coordinamento e sul costo computazionale. I principali schemi di comunicazione considerati nella letteratura MARL sono: (1) fully connected ($E = V \times V$), con complessità di comunicazione $O(N^2)$ ma massima condivisione di informazioni; (2) k-nearest neighbors, con complessità $O(kN)$ ma comunicazione limitata agli agenti fisicamente vicini; (3) grafo appreso (struttura E addestrabile), dove la rete apprende quali canali di comunicazione sono più utili per il task. L'ultimo approccio è implementato in MAGIC (Multi-Agent Graph Attention with Inductive Communication, Niu et al. 2021) tramite un meccanismo di attenzione che pondera i messaggi in arrivo: $h^i_{l+1} = \mathrm{GRU}(h^i_l, \Sigma_{j \in N(i)} \alpha_{ij}^{(l)} h^j_l)$, dove $\alpha_{ij}^{(l)} = \mathrm{softmax}_j(q^i_l \cdot k^j_l / \sqrt{d_k})$ sono i pesi di attenzione calcolati dai query-key dell'agente i sui vicini j.

Per sistemi robotici in ambienti dinamici, la topologia del grafo di comunicazione deve adattarsi alla posizione degli agenti: un robot nella posizione p_1 a distanza r_1 da un robot in p_2 dovrebbe comunicare se $\|p_1 - p_2\| \leq r_{comm}$ (range di comunicazione), producendo un grafo dinamico che cambia ad ogni timestep. La GNN dinamica (Dynamic GNN, EvolveGCN, Pareja et al. 2020) gestisce questo scenario aggiornando i pesi della GNN ad ogni step tramite una RNN che osserva la struttura del grafo corrente: $W_l(t+1) = \mathrm{RNN}_l(W_l(t), A(t))$, dove $A(t)$ è la matrice di adiacenza al tempo t. Questo approccio è stato applicato con successo al coordinamento di droni in formazione (Tolstaya et al. 2020) e può essere adattato alla coordinazione di flotte di robot umanoidi in spazi condivisi.

La connessione con i Capitoli precedenti si manifesta nella struttura equivariante: le GNN di coordinamento sono equivarianti al gruppo di simmetria del problema, che include le permutazioni degli agenti (§6.3). Questa equivarianza garantisce che la policy di coordinamento sia coerente rispetto all'etichettatura arbitraria degli agenti — una proprietà

fondamentale per la scalabilità: una policy addestrata su N agenti può essere applicata a N′ ≠ N agenti senza re-training, purché la struttura del grafo si adatti. Questo tipo di generalizzazione zero-shot nel numero di agenti è stato dimostrato per il problema di coverage control (Tolstaya et al. 2019) ed è una delle promesse principali dell'approccio GNN per la robotica multi-agente.

Appendice A — Notazione e Convenzioni Matematiche

La presente appendice raccoglie in modo sistematico le notazioni, le convenzioni e le abbreviazioni utilizzate nel corpo della monografia. Si adottano le convenzioni standard della letteratura sulla robotica matematica e sull'apprendimento automatico, con le sole deviazioni esplicitamente indicate.

A.1 Algebre di Lie e Gruppi di Lie

Nel testo si utilizzano i seguenti gruppi di Lie e le corrispondenti algebre:

SO(3) denota il gruppo speciale ortogonale tridimensionale, ossia l'insieme delle matrici $R \in \mathbb{R}^{3\times 3}$ con $R^{\mathsf{T}} R = I$ e $\det R = +1$. La corrispondente algebra di Lie $\mathfrak{so}(3)$ è lo spazio delle matrici antisimmetriche 3×3. La mappa esponenziale $\exp: \mathfrak{so}(3) \to SO(3)$ è data dalla formula di Rodrigues: $\exp(\theta\, \hat{k}^{\times}) = I + \sin\theta\, \hat{k}^{\times} + (1 - \cos\theta)\, \hat{k}^{\times 2}$, con $\hat{k}^{\times}$ la hat-map del vettore unitario $\hat{k}$.

SE(3) denota il gruppo speciale euclideo, prodotto semidiretto $SO(3) \ltimes \mathbb{R}^3$. Un elemento $T \in SE(3)$ è rappresentato dalla matrice 4×4 $[R,\ t;\ 0,\ 1]$ con $R \in SO(3)$ e $t \in \mathbb{R}^3$. L'algebra $\mathfrak{se}(3) \cong \mathbb{R}^6$ è parametrizzata dai twist $\xi = (\omega,\ v)^{\mathsf{T}}$ dove $\omega \in \mathbb{R}^3$ è la velocità angolare e $v \in \mathbb{R}^3$ è la velocità lineare del frame di corpo.

$\log: G \to \mathfrak{g}$ denota il logaritmo di gruppo, inverso dell'esponenziale nelle carte locali. Per $SO(3)$: $\log(R) = \theta/(2\sin\theta)\,(R - R^{\mathsf{T}})$ con $\theta = \arccos((\operatorname{tr} R - 1)/2)$. Per $SE(3)$: la formula di Rodrigues estesa fornisce $\log(T) = (\omega,\ v)$ con $\omega = \log_{SO(3)}(R)$ e $v = (I - R)/|\omega|^2\ \omega \times t + \omega\omega^{\mathsf{T}} t/|\omega|^2$ per $|\omega| \neq 0$.

A.2 Notazione Tensore e Operatori Differenziali

Le lettere minuscole in grassetto (es. $\mathbf{x}$, $\mathbf{q}$, $\boldsymbol{\tau}$) denotano vettori colonna di $\mathbb{R}^n$. Le lettere maiuscole in grassetto (es. $\mathbf{M}$, $\mathbf{J}$, $\mathbf{A}$) denotano matrici. Il prodotto esterno di due vettori u, v si scrive uv^{T}. Il prodotto di Kronecker è $\otimes$. Il prodotto di Hadamard (elemento per elemento) è $\odot$.

La hat-map $(\cdot)^\times: \mathbb{R}^3 \to so(3)$ è definita da $a^\times b = a \times b$ (prodotto vettoriale). In coordinate: per $a = (a_1, a_2, a_3)^T$, la matrice $a^\times$ è antisimmetrica con elementi $(a^\times)_{12} = -a_3$, $(a^\times)_{13} = a_2$, $(a^\times)_{23} = -a_1$. La mappa inversa è vee: $so(3) \to \mathbb{R}^3$.

Gli operatori differenziali adottati: $\nabla_x f$ denota il gradiente di f rispetto a x; $H_x f \equiv \nabla^2_x f$ è la matrice Hessiana; $J_x f$ è la Jacobiana (matrice delle derivate prime per funzioni vettoriali). La derivata di Lie di f lungo il campo vettoriale g è $L_g f = \nabla f \cdot g$. Per sistemi dinamici $\dot{x} = f(x) + g(x)u$, il grado relativo rispetto all'output $y = h(x)$ è il minimo k tale che $L_g L_f^{k-1} h(x) \neq 0$.

A.3 Notazione per Sistemi Probabilistici e Apprendimento

$p(x)$ denota la densità di probabilità (pdf) della variabile aleatoria X. $p(y|x)$ è la densità condizionale. $\mathbb{E}_{X\sim p}[f(X)]$ è il valore atteso di f(X) rispetto alla distribuzione p. Var(X) è la varianza scalare; Cov(X) la matrice di covarianza. $N(\mu, \Sigma)$ è la distribuzione gaussiana con media μ e covarianza Σ.

La divergenza di Kullback-Leibler tra le distribuzioni p e q è $D_{KL}(p \| q) = \int p(x) \log[p(x)/q(x)]\, dx$. La distanza di Wasserstein-p tra μ e ν è $W_p(\mu,\nu) = (\inf_{\gamma\in\Gamma(\mu,\nu)} \int \|x-y\|^p \, d\gamma)^{1/p}$. La Maximum Mean Discrepancy con kernel k è $MMD^2(p,q) = \mathbb{E}_{x,x'\sim p}[k(x,x')] - 2\mathbb{E}_{x\sim p,y\sim q}[k(x,y)] + \mathbb{E}_{y,y'\sim q}[k(y,y')]$.

A.4 Acronimi e Abbreviazioni

ACT Action Chunking with Transformers

ADMM Alternating Direction Method of Multipliers

AdEx Adaptive Exponential Integrate-and-Fire

BC Behavioral Cloning

CBF Control Barrier Function

CLF Control Lyapunov Function

CMM Centroidal Momentum Matrix

CNF Continuous Normalizing Flow

CoM Center of Mass

CoP Center of Pressure

CRBA Composite Rigid Body Algorithm

CVAE Conditional Variational AutoEncoder

DAE Differential-Algebraic Equation

DANN Domain-Adversarial Neural Network

DDPM Denoising Diffusion Probabilistic Model

DDIM Denoising Diffusion Implicit Models

DPO Direct Preference Optimization

EKF Extended Kalman Filter

ELBO Evidence Lower BOund

FEM Finite Element Method

FGSM Fast Gradient Sign Method

GAIL Generative Adversarial Imitation Learning

GRU Gated Recurrent Unit

HJI Hamilton-Jacobi-Isaacs

HQP Hierarchical Quadratic Program

HZD Hybrid Zero Dynamics

IL Imitation Learning

IK Inverse Kinematics

IMU Inertial Measurement Unit

K-FAC Kronecker-Factored Approximate Curvature

LCP Linear Complementarity Problem

LIF Leaky Integrate-and-Fire

LIPM Linear Inverted Pendulum Model

LoRA Low-Rank Adaptation

LQR Linear Quadratic Regulator

LTD Long-Term Depression

LTP Long-Term Potentiation

MHA Multi-Head Attention

MPC Model Predictive Control

MPM Material Point Method

NFE Number of Function Evaluations

OSQP Operator Splitting Quadratic Program

PD Proporzionale-Derivativo

PDDL Planning Domain Definition Language

PGD Projected Gradient Descent

PPO Proximal Policy Optimization

QP Quadratic Program

RLHF Reinforcement Learning from Human Feedback

RNEA Recursive Newton-Euler Algorithm

RoPE Rotary Position Embedding

RSSM Recurrent State Space Model

SE(2) Special Euclidean group in 2D

SE(3) Special Euclidean group in 3D

SEA Series Elastic Actuator

SLIP Spring-Loaded Inverted Pendulum

SDE Stochastic Differential Equation

SRM Spike Response Model

STDP Spike-Timing Dependent Plasticity

TAMP Task and Motion Planning

TD Temporal Difference

TRPO Trust Region Policy Optimization

VC Vapnik-Chervonenkis

ViT Vision Transformer

VLA Vision-Language-Action

VLM Vision-Language Model

VQ-VAE Vector Quantized Variational AutoEncoder

WBC Whole-Body Control

ZMP Zero Moment Point

Appendice B — Algoritmi e Pseudocodici

Questa appendice presenta in forma algoritmica compatta le procedure computazionali descritte nella monografia. La notazione segue le convenzioni standard dello pseudocodice con tipizzazione implicita; la complessità computazionale è indicata in notazione O-grande.

B.1 Algoritmo RNEA (Recursive Newton-Euler)

Input: configurazione η, velocità $\dot{\eta}$, accelerazione $\ddot{\eta} = 0$ (per calcolo termini non lineari), struttura cinematica ad albero con N link. Output: vettore $h(\eta, \dot{\eta}) = C(\eta,\dot{\eta})\dot{\eta} + g(\eta)$.

Algoritmo RNEA(η, $\dot{\eta}$, $\ddot{\eta}$):

Passata in avanti (i = 1, ..., N):

$V_i \leftarrow {}^{i}X_{i-1}\, V_{i-1} + S_i\, \dot{q}_i$

$\dot{V}_i \leftarrow {}^{i}X_{i-1}\, \dot{V}_{i-1} + S_i\, \ddot{q}_i + \dot{S}_i\, \dot{q}_i$

Passata all'indietro (i = N, ..., 1):

$F_i \leftarrow I_i^c\, \dot{V}_i + V_i \times^* I_i^c\, V_i - {}^{i}X_0\, f_i^{ext}$

$\tau_i \leftarrow S_i^T\, F_i$

$F_{i-1} \leftarrow F_{i-1} + {}^{i}X_{i-1}^T\, F_i$

return τ // = $h(\eta, \dot{\eta})$ per $\ddot{\eta} = 0$

Complessità: O(n) — circa 60n operazioni floating-point

B.2 Algoritmo CRBA (Composite Rigid Body)

Input: configurazione η. Output: matrice di inerzia $M(\eta) \in \mathbb{R}^{(6+n)\times(6+n)}$. Complessità: $O(n^2)$ in generale, $O(nd)$ per alberi con profondità $d \ll n$.

Algoritmo CRBA(η):

// *Inizializzazione inerzie composite*

Per i = 1, ..., N: $\hat{I}_i \leftarrow I_i^c$

// *Passata all'indietro: accumulo inerzie composite*

Per i = N, ..., 1:

$\hat{I}_{parent(i)} \leftarrow \hat{I}_{parent(i)} + {}^{i}X_{parent}^{T}\, \hat{I}_i\, {}^{i}X_{parent}$

// Costruzione colonne di M

Per j = 1, ..., n:

$F \leftarrow \hat{I}_j\, S_{\curvearrowright}$

$M_{j,j} \leftarrow S_{\curvearrowright}^{T}\, F$

$i \leftarrow parent(j)$

While i ≠ base:

$F \leftarrow {}^{i}X_{parent}^{T}\, F$

$M_{i,j} \leftarrow S_i^{T}\, F;\;\; M_{j,i} \leftarrow M_{i,j}$

$i \leftarrow parent(i)$

return M

B.3 Algoritmo di Orin-Goswami per la CMM

Input: configurazione η, velocità $\dot{\eta}$. Output: Centroidal Momentum Matrix $A(\eta) \in \mathbb{R}^{6\times(6+n)}$ e momento centroidale $h = A(\eta)\,\dot{\eta}$. Complessità: O(n).

Algoritmo CMM_OrinGoswami(η, η̇):

// Passata in avanti: calcolo velocità frame centroidale

Calcola p_o (posizione CoM del sistema)

Per i = 1, ..., N:

$v_{c,i} \leftarrow v_{c,parent(i)} + \omega_{parent(i)} \times r_{i,parent} + \omega_i \times r_{c,i}$

$\omega_i \leftarrow \omega_{parent(i)} + R_i\, \hat{z}_i\, \dot{q}_i$

// Passata all'indietro: aggregazione contributi

$A \leftarrow 0_{6\times(6+n)}$

Per i = N, ..., 1:

$h_i \leftarrow [m_i\, v_{c,i};\;\; I_i^{c}\, \omega_i + m_i\, (p_{c,i} - p_o) \times v_{c,i}]$

$A_i \leftarrow [m_i\, J_{v,i};\;\; I_i^{c}\, J_{\omega,i} + m_i\, [p_{c,i} - p_o]\times J_{v,i}]$

$A \leftarrow A + A_i$

return A, h = A η̇

B.4 Algoritmo QP Gerarchico (HQP) con Active-Set

Input: sequenza di task {J◎, ẋ□}_{k=1..m}, vincoli di disuguaglianza C x ≥ d. Output: soluzione gerarchica x* ∈ $\mathbb{R}^{n_x}$. Complessità per ciclo: $O(n_x^2 \cdot n_o)$.

Algoritmo HQP({J⌯, ẋ□}, C, d):

x_p ← 0; Z ← I_{n_x} // soluzione particolare e null-space iniziale

A_eq ← []; b_eq ← []

// Warm-start: aggiorna insieme attivo dall'istante precedente

Per k = 1, ..., m:

A_red ← J⌯ Z

b_red ← ẋ□ − J⌯ x_p

// Minimi quadrati nel sottospazio residuo

[Q_k, R_k] ← QR(A_red)

y ← R_k⁻¹ Q⌯ᵀ b_red (pseudoinverso)*

*x_p ← x_p + Z y**

// Aggiorna null-space

Z ← Z · null(A_red) // proiettore nel null-space di J⌯ Z

// Applica vincoli di disuguaglianza con active-set method

Fino a convergenza KKT:

Risolvi QP di uguaglianza con insieme attivo corrente

Verifica condizioni KKT; aggiorna insieme attivo

*return x**

B.5 Algoritmo Flow Matching Condizionale (CFM)

Addestramento: dato dataset di azioni $\{a^0_i\}$ e osservazioni $\{o_i\}$. Campionamento: dato contesto o, genera azione a^0 integrando l'ODE. Numero tipico di passi NFE: 10–50.

Training CFM(θ):

Per ogni batch (a^0, o):

ε ~ N(0, I) // rumore sorgente

t ~ U[0,1]

$a_t \leftarrow (1-t)\varepsilon + t \cdot a^0$ // traiettoria lineare

loss ← $\|v_\theta(a_t, t, o) - (a^0 - \varepsilon)\|^2$

θ ← θ − α ∇_θ loss

Campionamento CFM(o):

a_0 ~ N(0, I)

// Integrazione ODE con Eulero o Runge-Kutta

Per t = 0, dt, 2dt, ..., 1−dt:

a_{t+dt} ← a_t + dt · v_θ(a_t, t, o)

return a_1 // azione generata

B.6 Algoritmo PDDLStream per TAMP

Input: dominio PDDL, stato iniziale s_0, goal g, stream $S = \{s_1,...,s_\circledcirc\}$. Output: piano geometrico fattibile $\pi = (a_1,...,a_\circledcirc)$ se esiste. Completezza probabilistica garantita.

Algoritmo PDDLStream(D_PDDL, s_0, g, S):

D_ext ← D_PDDL // dominio esteso con campioni

Ripeti:

π_cand ← TaskPlanner(D_ext, s_0, g)

Se π_cand = ∅: return INFATTIBILE

// Verifica fattibilità geometrica

Per ogni azione a_i in π_cand:

Se a_i richiede parametri continui non ancora campionati:

Val ← SampleStream(a_i, S)

Se Val = ∅: break // fallimento locale

D_ext ← D_ext ∪ {certificati(Val)}

Se tutti i parametri trovati:

Se VerificaTraiettorie(π_cand): return π_cand

// Ripeti con più campioni

Appendice C — Risultati di Complessità e Dimostrazioni Supplementari

Questa appendice raccoglie risultati di complessità computazionale e dimostrazioni supplementari che completano il corpo principale della monografia, offrendo approfondimenti tecnici utili a chi intende implementare o estendere gli algoritmi presentati.

C.1 Complessità dei Principali Algoritmi

La tabella seguente riassume la complessità computazionale per i principali algoritmi discussi nella monografia. n indica il numero di gradi di libertà, k il numero di contatti attivi, N_b il numero di link, d la profondità media dell'albero cinematico, e n_x la dimensione del vettore di decisione del QP.

Algoritmo	**Complessità**	**Note**
RNEA	O(n)	~60n FLOP; ottimale per sistemi ad albero
CRBA	$O(n^2)$	ridotto a O(nd) per alberi bilanciati
CMM Orin-Goswami	O(n)	passata avanti + passata indietro
HQP gerarchico	$O(n_x^2 \cdot m)$	m = numero task; warm-start riduce a $O(n_x^2)$
OSQP (solver QP)	O(n^{1.5}) per iterazione	50–200 iterazioni tipiche per n ≤ 50
qpOASES (active-set)	$O(n_x^3)$ nel caso peggiore	1–5 iterazioni con warm-start
Flow Matching sampling	O(NFE · d)	NFE = 10–50; d = dimensione azione
DDPM sampling	O(K · d)	K = 100–1000 passi; DDIM riduce a ~50
Transformer forward	$O(n^2\_tok \cdot d\_model)$	n_tok = lunghezza sequenza
ViT forward (singola immagine)	$O(N^2 \cdot d)$	N = numero patch = $(H \cdot W)/P^2$
Sinkhorn (trasporto ottimale)	$O(n^2/\varepsilon)$ iter.	n = campioni; ε = regolarizzazione
EKF aggiornamento	$O(n\_\varphi^2)$	n_φ = dim. parametri da stimare
CRBA-ident. (regressore Y)	O(n · 10 N_b)	lineare in N_b per struttura ad albero
LCP Delassus (Lemke)	$O(k^3)$	k = contatti attivi; tipicamente k ≤ 8

C.2 Dimostrazione: Antisimmetria di $\dot{M} - 2C$ via Christoffel

Si dimostra qui in modo esplicito e autocontenuto la Proposizione 1.1 della monografia, mostrando che la scelta della matrice di Coriolis tramite i simboli di Christoffel di seconda specie implica l'antisimmetria di $N = \dot{M} - 2C$, con conseguenze sulla passività del sistema.

> Definizione. Per una metrica riemanniana g_{ij}(q) ≡ M_{ij}(η) sulla varietà di configurazione Q, i simboli di Christoffel della seconda specie sono $\Gamma_{i}\square\square(\eta) = \tfrac{1}{2}(\partial M_{ij}/\partial \eta_k + \partial M_{ik}/\partial \eta_j - \partial M_{jk}/\partial \eta_i)$. La matrice di Coriolis è C_{ij}(η,η̇) = Σ⊚ $\Gamma_{i}\square\square$ η̇□.
>
> Calcolo esplicito di N_{ij}. Derivando M_{ij}(η(t)) rispetto a t: $\dot{M}_{ij} = \Sigma_{\circledcirc} (\partial M_{ij}/\partial \eta_k)\ \dot{\eta}\square$. Quindi $N_{ij} = \dot{M}_{ij} - 2C_{ij} = \Sigma_{\circledcirc} (\partial M_{ij}/\partial \eta_k)\ \dot{\eta}\square - \Sigma_{\circledcirc} (\partial M_{ij}/\partial \eta_k + \partial M_{ik}/\partial \eta_j - \partial M_{jk}/\partial \eta_i)\ \dot{\eta}\square = \Sigma_{\circledcirc} (\partial M_{jk}/\partial \eta_i - \partial M_{ik}/\partial \eta_j)\ \dot{\eta}\square$.
>
> Antisimmetria. Calcolando N_{ji}: $N_{ji} = \Sigma_{\circledcirc} (\partial M_{ik}/\partial \eta_j - \partial M_{jk}/\partial \eta_i)\ \dot{\eta}\square = -N_{ij}$. Quindi $N^{T} = -N$, ovvero N è antisimmetrica. □

Corollario (Passività). Moltiplicando l'equazione del moto per $\dot{\eta}^{T}$ a sinistra: $\dot{\eta}^{T} M \ddot{\eta} + \dot{\eta}^{T} C \dot{\eta} + \dot{\eta}^{T} g = \dot{\eta}^{T}(S\tau + Jc^{T}\lambda)$. Il termine $\dot{\eta}^{T} N \dot{\eta} = 0$ per antisimmetria, quindi $d/dt(T + V) = \dot{\eta}^{T}(S\tau + Jc^{T}\lambda)$: la potenza iniettata equivale al rateo di variazione dell'energia meccanica, che è la definizione di sistema passivo rispetto alla coppia ingresso.

C.3 Derivazione del Bound di Complessità Campionaria per BC

Si deriva qui il bound H^2 del Behavioral Cloning (Teorema 12.1) tramite il lemma di performance di Kakade-Langford. Il risultato mostra che l'errore di imitazione si accumula quadraticamente con l'orizzonte di pianificazione H.

> Lemma di performance (Kakade-Langford). Per due policy π_1 e π_2 su un MDP con ricompensa massima R_max e orizzonte H, il value gap soddisfa: $V(\pi_1) - V(\pi_2) \le H \cdot R_{max} \cdot E_{s\sim d^{\pi_2}}[D_{TV}(\pi_1(\cdot|s), \pi_2(\cdot|s))]$, dove d^{π_2} è la distribuzione di occupazione degli stati sotto π_2.

Applicazione al BC. Sia π_E la policy esperta e π_n la policy appresa da n dimostrazioni. Per il BC, π_n minimizza l'errore di classificazione sugli stati dell'esperto: $\varepsilon\_n = E\_{s\sim d^{\pi\_E}}[1(\pi\_n(s) \neq \pi\_E(s))]$. Il problema è che π_n viene valutata sugli stati visitati dalla stessa π_n, non dall'esperto: la distribuzione d^{π_n} differisce da d^{π_E}.

Stima del distributional shift. Per ogni passo $t \in \{1,...,H\}$, l'errore su d^{π_n} è al più $E\_{s\sim d^{\pi\_n}}[D\_TV] \leq \varepsilon\_n + \delta\_t$ dove δ_t cresce con il distributional shift. Sotto l'assunzione che il singolo errore di imitazione ε_n si propaghi uniformemente: $V(\pi\_E) - V(\pi\_n) \leq H \cdot R\_max \cdot \Sigma\_{t=1}^{H} \varepsilon\_n = H^2 \cdot R\_max \cdot \varepsilon\_n$.

Interpretazione. Il termine H^2 indica che raddoppiare l'orizzonte richiede quadruplicare il numero di dimostrazioni per mantenere la stessa precisione. DAgger rimuove il distributional shift campionando dati sugli stati di π_n, riducendo il bound a $H \cdot \varepsilon\_n$. □

C.4 Proprietà Spettrali della Matrice di Delassus

La matrice di Delassus $H = J\_a\, M^{-1}\, J\_a^T \in \mathbb{R}^{k\times k}$ governa l'esistenza e unicità della soluzione del LCP di contatto. Si riportano qui le principali proprietà spettrali e le loro implicazioni per la simulazione numerica.

Proprietà 1 (Semidefinita positiva). H è semidefinita positiva per ogni configurazione: per ogni $v \in \mathbb{R}^k$, $v^T H v = v^T J\_a M^{-1} J\_a^T v = \|M^{-1/2} J\_a^T v\|^2 \geq 0$. Il nucleo di H coincide con il nucleo di J_a^T: se J_a ha rango pieno per righe ($k \leq n$), H è definita positiva con $\lambda\_min(H) \geq \lambda\_min(M^{-1})\, \sigma\_min^2(J\_a) > 0$.

Proprietà 2 (Numero di condizione). Il numero di condizione $\kappa(H) = \lambda\_max(H)/\lambda\_min(H)$ cresce con il numero di contatti e con la distanza dalle singolarità cinematiche. In configurazioni vicine a singolarità (J_a quasi rank-deficient), $\lambda\_min(H) \to 0$ e $\kappa(H) \to \infty$, rendendo il LCP numericamente mal condizionato.

Proprietà 3 (Condizionamento regolarizzato). La regolarizzazione soft-contact $H\_\varepsilon = H + \varepsilon I$ con $\varepsilon > 0$ piccolo garantisce $\lambda\_min(H\_\varepsilon) \geq \varepsilon$,

controllando il numero di condizione $\kappa(H_\varepsilon) \leq \|H\|_2/\varepsilon + 1$. Questo è il fondamento teorico del contact softening adottato in MuJoCo e Genesis.

Appendice D — Dati di Benchmark e Risultati Sperimentali

Questa appendice riporta i benchmark di riferimento, i dataset e i risultati sperimentali citati nel corso della monografia. I dati sono organizzati per categoria tematica e corredati di riferimenti alla letteratura primaria.

D.1 Benchmark per Sistemi Umanoidi

Piattaforma	DOF	Massa (kg)	Altezza (m)	Payload (kg)	WBC @ (Hz)
Figure 02 (Apptronik)	16+	75	1.67	20	500
Tesla Optimus Gen-2	28	57	1.73	20	1000
Boston Dynamics Atlas (elettrico)	28	89	1.50	15	1000
Agility Robotics Digit v4	20	65	1.60	16	500
Unitree H1	19	47	1.80	30	500
Fourier GR-1	44	55	1.65	10	1000

D.2 Dataset per l'Apprendimento per Imitazione

I seguenti dataset sono i principali riferimenti per la valutazione dei modelli VLA e delle policy di imitazione discussi nella monografia.

Open X-Embodiment (OXE) — 527 000 traiettorie, 23 robot, 22 dataset. Padalkar et al. 2023.

LIBERO — 130 compiti, 3 split tematici, 10 demo/task. Liu et al. 2023.

DROID — 76 000 traiettorie, manipolatore Franka, scene domestiche. Khazatsky et al. 2024.

RoboVerse — 1 M+ traiettorie simulate, 5 ambienti, 100+ task. Wang et al. 2024.

BridgeData V2 — 60 000 traiettorie, kitchen manipulation. Walke et al. 2023.

ALOHA bimanual — 50–200 demo/task, cucina e ufficio. Zhao et al. 2023.

CMU Play-Fusion — 1 200 ore di dati play teleoperation. Shridhar et al. 2023.

D.3 Risultati sui Benchmark VLA

Si riportano i risultati di successo task (%) sui principali benchmark per i modelli VLA discussi nei Capitoli 5–7. I valori sono tratti dalle pubblicazioni originali; "novel obj." indica task con oggetti non visti durante il training.

Modello	Parametri	OXE novel obj. (%)	LIBERO-90 (%)	Note
RT-1	35 M	53	—	Backbone ViT-base
RT-2 (PaLM-E)	55 B	67	—	Chain-of-thought abilitato
OpenVLA	7 B	60	71	Fine-tuning su Llama-2
π_0 (Physical Intelligence)	3 B	73	82	Flow matching actions
π_0.5	3 B	79	85	Cross-embodiment + casa aperta
Octo	93 M	58	78	Transformer diffusion head
RoboVLMs (ensemble)	7 B+	75	84	Multi-backbone fusion

Appendice F — Tavola delle Principali Formule Supplementari

Si raccolgono qui le equazioni chiave introdotte nei capitoli supplementari, organizzate per riferimento rapido.

SF.1 PINN per Identificazione (Cap. S1)

L_PINN = L_data + lambda_phys * L_phys + lambda_reg * L_reg

L_phys = || M(pi_hat) - M(pi_hat)^T ||_F^2 + max(0, -lambda_min(M(pi_hat)))

SF.2 Controllo di Impedenza (Cap. S2)

M_d * x_ddot_e + D_d * x_dot_e + K_d * x_e = F_ext

tau = Y(eta, eta_dot, eta_ddot_r) pi + J^T (F_d - K_d x_e - D_d x_dot_e)

SF.3 RoboPoint (Cap. S3)

p* = argmax_p P_VLM(p | I, g)

a* = argmax_a P_action(a | I, p*, g)

SF.4 World Model Causale (Cap. S4)

P(s_{t+1} | do(a_t)) != P(s_{t+1} | a_t) (effetto causale vs correlazione)

SF.5 Complessita Campionaria Bimanuale (Cap. S5)

V(pi_E) - V(pi_n) <= H^2 * epsilon_n + H^2 * I(a^L; a^R | o) / n

SF.6 Augmentazione SE(3) (Cap. S5)

I'_t = Render(R * scene_t + t)

a'_t = R * a_t, p'_t = R * p_t + t

SF.7 Pianificazione Differenziabile (Cap. S6)

theta* = argmin_{theta} L_task(execute(theta)) + lambda * L_smooth(theta)

SF.8 Reward dalla Mappa di Poincare (Cap. S7)

R_Poincare(pi) = -lambda * max(0, rho(DP_pi(x*)) - rho_target)

SF.9 Complessita Intrinseca (Cap. S8)

n* = Omega((d_int / epsilon^2) * log(1/delta))

B* >= d_int/2 * log_2(sigma_action / epsilon) (bit minimi per token azione)

SF.10 Composizione CBF (Cap. S9)

h_AND(x) = -1/mu * log(e^{-mu*h_1} + e^{-mu*h_2}) (soft AND)

h_OR(x) = 1/mu * log(e^{mu*h_1} + e^{mu*h_2}) (soft OR)

SF.11 Distillazione SNN (Cap. S10)

L_KD_spike = alpha*L_task + (1-alpha) * sum_l ||h_l^S - W_l*h_l^T||^2

E_S1_Loihi ~ N_neurons * r_spike * E_AC * f_ctrl ~ 0.5 W @ 1 kHz

Appendice E — Framework Software Open-Source di Riferimento

Questa appendice fornisce una panoramica sistematica dei principali framework software open-source che consentono di replicare e estendere i risultati presentati nella monografia. Per ciascun framework sono indicati: il capitolo di riferimento nella monografia, le funzionalita principali, i requisiti hardware minimi, e un esempio di utilizzo minimo.

SE.1 Whole-Body Control e Dinamica Multi-Corpo

Crocoddyl

Riferimento: Cap. 2.5 (DDP, iLQR), Cap. 3.6 (ottimizzazione di traiettoria).

Crocoddyl (Mastalli et al., 2022) e il framework open-source di riferimento per il Differential Dynamic Programming con contatti. Supporta sistemi floating-base, contatti unilaterali, e WBC gerarchico. Implementa FDDP (Feasibility-Driven DDP) e Box-FDDP.

```
pip install crocoddyl
```

Funzionalita principali: modelli di dinamica per manipolatori seriali e floating-base, supporto per URDF/MJCF, contatti con LCP softened, interfaccia Python.

Hardware minimo: CPU 4-core, 8 GB RAM. Per training RL: GPU NVIDIA con CUDA 11+.

Pinocchio

Riferimento: Cap. 1.1–1.5 (RNEA, CRBA, CMM).

Pinocchio e la libreria C++/Python per la cinematica e dinamica dei robot che implementa in modo ottimale RNEA, CRBA e CMM. E il backend computazionale di Crocoddyl e di molti altri framework.

```
pip install pin
```

Funzionalita principali: cinematica diretta e inversa, RNEA O(n), CRBA O(n^2), Jacobiani analitici e geometrici, supporto SE(3) nativo.

MuJoCo e MuJoCo MJX

Riferimento: Cap. 9.1–9.3 (simulazione differenziabile, time-stepping).

MuJoCo (Todorov et al., 2012) e il simulatore fisico di riferimento per la ricerca in RL robotico. MuJoCo MJX e la variante JAX per GPU con autodifferenziazione.

```
pip install mujoco   # CPU
pip install mujoco-mjx   # GPU/TPU con JAX
```

Funzionalita principali: simulazione con contatti softened, integrazione semi-implicita, supporto MJCF, simulazione differenziabile in MJX, parallelizzazione tramite vmap.

IsaacLab

Riferimento: Cap. 9.2 (RL massivamente parallelo).

IsaacLab e il framework NVIDIA per RL robotico massivamente parallelo basato su Isaac Sim (PhysX 5). Supporta migliaia di ambienti paralleli su singola GPU.

```
# Richiede NVIDIA GPU, Isaac Sim, e Python >= 3.10
pip install isaaclab
```

Funzionalita principali: 10^4 ambienti paralleli, sensori fisici realistici, curriculum learning automatico, import URDF/MJCF.

SE.2 Apprendimento per Imitazione e Policy

LeRobot

Riferimento: Cap. 11 (ACT, Diffusion Policy, policy di imitazione).

LeRobot (Hugging Face, 2024) e il framework open-source piu completo per l'imitation learning robotico. Include implementazioni di ACT, Diffusion Policy, TD-MPC2, e architetture VLA con interfaccia standardizzata.

```
pip install lerobot
```

Funzionalita principali: dataset ALOHA e Open X-Embodiment, training ACT e Diffusion Policy, support per ALOHA 2 e hardware Dynamixel, valutazione standardizzata su benchmark.

FROST

Riferimento: Cap. 3.6 (DDP ibrido per HZD), Cap. 3.3 (ottimizzazione cicli HZD).

FROST (Fast Robot Optimization and Simulation Toolkit) e il framework MATLAB/Julia per la sintesi di controllori HZD e l'ottimizzazione di traiettorie per robot bipedi.

Funzionalita principali: ottimizzazione dei coefficienti di Bezier, calcolo della mappa di Poincare, stabilita orbitale tramite IPOPT, supporto per robot bipedi con modelli URDF.

OpenVLA

Riferimento: Cap. 5.1–5.4 (architettura VLA, LoRA fine-tuning).

OpenVLA e il modello VLA open-source da 7B parametri basato su LLaMA-2, disponibile su Hugging Face. Supporta fine-tuning con LoRA, DPO, e RLHF robotico.

```
pip install transformers accelerate
# from huggingface_hub import hf_hub_download
```

```
# model = AutoModelForCausalLM.from_pretrained("openvla/openvla-7b")
```

SE.3 Simulazione Differenziabile e World Models

Genesis

Riferimento: Cap. 9.4 (corpi deformabili, FEM, MPM).

Genesis (CMU, 2024) e il simulatore fisico differenziabile che supporta corpi rigidi, elastici (FEM), granulari (MPM), e fluidi (SPH) in un framework unificato con autodifferenziazione JAX.

```
pip install genesis-world
```

Funzionalita principali: simulazione multi-materiale differenziabile, import URDF/MJCF, renderer fotorealistico integrato, Python API.

Dreamer V3 (DreamerV3-torch)

Riferimento: Cap. 8 (RSSM, world model).

Implementazione PyTorch di Dreamer V3 con supporto per task di manipolazione.

```
git clone https://github.com/danijar/dreamerv3
```

```
pip install -r requirements.txt
```

Funzionalita principali: RSSM con variabili categoriche, trasformazione symlog, actor-critic su traiettorie immaginate, supporto MuJoCo e Atari.

SE.4 Sicurezza Certificata

CBFPY

Riferimento: Cap. 16 (CBF, QP unificato).

CBFPY e una libreria Python per la sintesi di controllori CBF con interfaccia a OSQP e qpOASES.

```
pip install cbfpy
```

Funzionalita principali: CBF di ordine 1 e 2, HOCBF, QP unificato CLF-CBF, supporto per sistemi continui e ibridi, visualizzazione dei safe set.

hj_reachability

Riferimento: Cap. 17 (HJI reachability, level set methods).

hj_reachability e una libreria JAX per la soluzione numerica dell'equazione HJI tramite level set methods con GPU acceleration.

```
pip install hj-reachability
```

Funzionalita principali: BRT computation, backward e forward reachability, deep HJI con reti neurali, import di sistemi MuJoCo.

SE.5 Multi-Agent RL

EPyMARL

Riferimento: Cap. 26 (MARL, QMIX, MAPPO).

EPyMARL (Extended PyMARL) e il framework di riferimento per il multi-agent reinforcement learning cooperativo con implementazioni di QMIX, MAPPO, CommNet e MADDPG.

```
git clone https://github.com/uoe-agents/epymarl
pip install -r requirements.txt
```

Funzionalita principali: ambienti SMAC e MPE, algoritmi QMIX/VDN/MAPPO/MADDPG, logging con Weights & Biases, benchmark standardizzati.

Bibliografia

I riferimenti bibliografici sono ordinati per ordine alfabetico del primo autore. Per articoli di conferenza si indica l'acronimo della venue; per preprint si indica arXiv con il numero identificativo. DOI e link permanenti sono disponibili sui repository istituzionali degli autori.

[Ahn2022] Ahn, M., Brohan, A., Brown, N., et al. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. arXiv:2204.01691, 2022.

[Albergo2023] Albergo, M. S., & Vanden-Eijnden, E. Building Normalizing Flows with Stochastic Interpolants. ICLR 2023.

[Ames2017] Ames, A. D., Xu, X., Grizzle, J. W., & Tabuada, P. Control Barrier Function Based Quadratic Programs for Safety Critical Systems. IEEE Transactions on Automatic Control 62(8), 2017.

[Amari1998] Amari, S.-I. Natural Gradient Works Efficiently in Learning. Neural Computation 10(2), 1998.

[Bansal2021] Bansal, S., Sidrane, L., & Tomlin, C. J. DeepReach: A Deep Learning Approach for Computing Reachable Sets. IEEE CDC, 2021.

[BenDavid2010] Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., & Vaughan, J. W. A Theory of Learning from Different Domains. Machine Learning 79(1–2), 2010.

[Blickhan1989] Blickhan, R. The Spring-Mass Model for Running and Hopping. Journal of Biomechanics 22(11–12), 1989.

[Brenier1991] Brenier, Y. Polar Factorization and Monotone Rearrangement of Vector-Valued Functions. Communications on Pure and Applied Mathematics 44(4), 1991.

[Brette2005] Brette, R., & Gerstner, W. Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity. Journal of Neurophysiology 94(5), 2005.

[Chang2020] Chang, Y.-C., Roohi, N., & Gao, S. Neural Lyapunov Control. NeurIPS 2020.

[Chi2023] Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B., & Song, S. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. RSS 2023.

[Chi2024] Chi, C., Xu, Z., Feng, S., Cousineau, E., Du, Y., Burchfiel, B., Tedrake, R., & Song, S. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. IJRR 2024.

[Cohen2018] Cohen, T. S., Geiger, M., Köhler, J., & Welling, M. Spherical CNNs. ICLR 2018.

[Cohen2019] Cohen, J., Rosenfeld, E., & Kolter, J. Z. Certified Adversarial Robustness via Randomized Smoothing. ICML 2019.

[Cottle1968] Cottle, R. W., & Dantzig, G. B. Complementary Pivot Theory of Mathematical Programming. Linear Algebra and its Applications 1(1), 1968.

[Cuturi2013] Cuturi, M. Sinkhorn Distances: Lightspeed Computation of Optimal Transport Distances. NeurIPS 2013.

[Dawson2023] Dawson, C., Qin, B., Gao, S., & Fan, C. Safe Control with Learned Certificates. IEEE RAL 2023.

[Dennis2020] Dennis, M., Jaques, N., Vinitsky, E., et al. Emergent Complexity and Zero-Shot Transfer via Unsupervised Environment Design. NeurIPS 2020.

[Dou2024] Dou, Z., et al. GROOT: Learning to Manipulate Novel Objects from Demonstrations with Goal-Conditioned Policy. CoRL 2024.

[Featherstone2008] Featherstone, R. Rigid Body Dynamics Algorithms. Springer, 2008. ISBN 978-0-387-74314-8.

[Ferreau2014] Ferreau, H. J., Kirches, C., Potschka, A., Bock, H. G., & Diehl, M. qpOASES: A Parametric Active-Set Algorithm for Quadratic Programming. Mathematical Programming Computation 6(4), 2014.

[Finn2017] Finn, C., Abbeel, P., & Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. ICML 2017.

[Follmer1988] Föllmer, H. Random Fields and Diffusion Processes. Lecture Notes in Mathematics 1362, Springer, 1988.

[Frantar2022] Frantar, E., Ashkboos, S., Hoefler, T., & Alistarh, D. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. ICLR 2023.

[Ganin2016] Ganin, Y., Ustunova, E., Ajakan, H., et al. Domain-Adversarial Training of Neural Networks. Journal of Machine Learning Research 17(1), 2016.

[Garrett2020] Garrett, C. R., Chitnis, R., Holladay, R., Kim, B., Silver, T., Kaelbling, L. P., & Lozano-Pérez, T. Integrated Task and Motion Planning. Annual Review of Control, Robotics, and Autonomous Systems 4, 2021.

[Gerstner2002] Gerstner, W., & Kistler, W. M. Spiking Neuron Models: Single Neurons, Populations, Plasticity. Cambridge University Press, 2002.

[Geyer2005] Geyer, H., Seyfarth, A., & Blickhan, R. Spring-Mass Running: Simple Approximate Solution and Application to Gait Stability. Journal of Theoretical Biology 232(3), 2005.

[Goodfellow2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. Generative Adversarial Networks. NeurIPS 2014.

[Goodfellow2015] Goodfellow, I. J., Shlens, J., & Szegedy, C. Explaining and Harnessing Adversarial Examples. ICLR 2015.

[GoogleDeepMind2025] Google DeepMind Robotics Team. Gemini Robotics: Bringing AI into the Physical World. Technical Report, 2025.

[Grizzle2003] Grizzle, J. W., Abba, G., & Plestan, F. Asymptotically Stable Walking for Biped Robots: Analysis via Systems with Impulse Effects. IEEE Transactions on Automatic Control 46(1), 2003.

[Gulrajani2017] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. Improved Training of Wasserstein GANs. NeurIPS 2017.

[Hafner2023] Hafner, D., Lillicrap, T., Norouzi, M., & Ba, J. Mastering Diverse Domains through World Models. arXiv:2301.04104, 2023.

[Hansen2024] Hansen, N., Su, H., & Wang, X. TD-MPC2: Scalable, Robust World Models for Continuous Control. ICLR 2024.

[Hejna2024] Hejna, J., Agarwal, A., Zhu, H., et al. OpenVLA: An Open-Source Vision-Language-Action Model. arXiv:2406.09246, 2024.

[Hereid2017] Hereid, A., & Ames, A. D. FROST: Fast Robot Optimization and Simulation Toolkit. IROS 2017.

[Ho2020] Ho, J., Jain, A., & Abbeel, P. Denoising Diffusion Probabilistic Models. NeurIPS 2020.

[Ho2016] Ho, J., & Ermon, S. Generative Adversarial Imitation Learning. NeurIPS 2016.

[Hoffmann2022] Hoffmann, J., Borgeaud, S., Mensch, A., et al. Training Compute-Optimal Large Language Models. NeurIPS 2022.

[Huang2022] Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., et al. Inner Monologue: Embodied Reasoning through Planning with Language Models. CoRL 2022.

[Ito1984] Ito, M. The Cerebellum and Neural Control. Raven Press, 1984.

[Kakade2002] Kakade, S. M. A Natural Policy Gradient. NeurIPS 2002.

[Kawasaki1991] Kawasaki, H., Berthier, T., & Angeles, J. A New Minimum Parameter Model of Robot Manipulators. IEEE Transactions on Robotics and Automation, 1991.

[Khazatsky2024] Khazatsky, A., Pertsch, K., Nair, S., et al. DROID: A Large-Scale In-The-Wild Robot Manipulation Dataset. arXiv:2403.12945, 2024.

[Lee2020] Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., & Hutter, M. Learning Quadrupedal Locomotion over Challenging Terrain. Science Robotics 5(47), 2020.

[Leonard2012] Léonard, C. A Survey of the Schrödinger Problem and Some of Its Connections with Optimal Transport. Discrete and Continuous Dynamical Systems 34(4), 2014.

[Liang2023] Liang, J., Huang, W., Xia, F., et al. Code as Policies: Language Model Programs for Embodied Control. ICRA 2023.

[Lipman2022] Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., & Le, M. Flow Matching for Generative Modeling. ICLR 2023.

[Liu2023] Liu, S., Zhu, H., Gao, F., et al. LIBERO: Benchmarking Knowledge Transfer in Lifelong Robot Learning. NeurIPS 2023.

[Liu2024] Liu, B., Jiang, Y., et al. LLM+P: Empowering Large Language Models with Optimal Planning Proficiency. arXiv:2304.11477, 2024.

[Ma2023] Ma, Y. J., Liang, W., Wang, G., Huang, D. A., et al. EUREKA: Human-Level Reward Design via Coding Large Language Models. arXiv:2310.12931, 2023.

[Maass1997] Maass, W. Networks of Spiking Neurons: The Third Generation of Neural Network Models. Neural Networks 10(9), 1997.

[Madry2018] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. ICLR 2018.

[Mansour2009] Mansour, Y., Mohri, M., & Rostamizadeh, A. Domain Adaptation: Learning Bounds and Algorithms. COLT 2009.

[Marsden2001] Marsden, J. E., & West, M. Discrete Mechanics and Variational Integrators. Acta Numerica 10, 2001.

[Mastalli2022] Mastalli, C., Budhiraja, R., Merkt, W., et al. Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control. ICRA 2022.

[Mitchell2005] Mitchell, I. M., Bayen, A. M., & Tomlin, C. J. A Time-Dependent Hamilton-Jacobi Formulation of Reachable Sets for Continuous Dynamic Games. IEEE Transactions on Automatic Control 50(7), 2005.

[Moreau1988] Moreau, J.-J. Unilateral Contact and Dry Friction in Finite Freedom Dynamics. CISM Courses and Lectures 302, Springer, 1988.

[Nagabandi2019] Nagabandi, A., Clavera, I., Liu, S., et al. Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning. ICLR 2019.

[Nagumo1942] Nagumo, M. Über die Lage der Integralkurven gewöhnlicher Differentialgleichungen. Proceedings of the Physico-Mathematical Society of Japan 24, 1942.

[Nichol2021] Nichol, A. Q., & Dhariwal, P. Improved Denoising Diffusion Probabilistic Models. ICML 2021.

[Oquab2023] Oquab, M., Darcet, T., Moutakanni, T., et al. DINOv2: Learning Robust Visual Features without Supervision. arXiv:2304.07193, 2023.

[Orin2013] Orin, D. E., & Goswami, A. Centroidal Momentum Matrix of a Humanoid Robot: Structure and Properties. IROS 2013.

[Padalkar2023] Padalkar, A., Pooley, A., Jain, A., et al. Open X-Embodiment: Robotic Learning Datasets and RT-X Models. arXiv:2310.08864, 2023.

[PhysicalIntelligence2024] Black, K., Brown, N., Driess, D., et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. arXiv:2410.24164, 2024.

[Rafailov2023] Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., & Finn, C. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. NeurIPS 2023.

[Radford2021] Radford, A., Kim, J. W., Hallacy, C., et al. Learning Transferable Visual Models From Natural Language Supervision. ICML 2021.

[Rocamonde2023] Rocamonde, J., Montesinos-Garcia, V., Perez-Higueras, N., et al. Vision-Language Models are Zero-Shot Reward Models for Reinforcement Learning. arXiv:2310.12921, 2023.

[Ross2011] Ross, S., Gordon, G., & Bagnell, D. A. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. AISTATS 2011.

[Schatzman1998] Schatzman, M. Numerical Integration of Dynamical Systems with Smooth Dependence. Mathematics of Computation, 1998.

[Schulman2015] Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. Trust Region Policy Optimization. ICML 2015.

[Schulman2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. Proximal Policy Optimization Algorithms. arXiv:1707.06347, 2017.

[Slotine1987] Slotine, J.-J. E., & Li, W. On the Adaptive Control of Robot Manipulators. Int. J. Robotics Research 6(3), 1987.

[Song2021] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., & Poole, B. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR 2021.

[Stellato2020] Stellato, B., Banjac, G., Goulart, P., Bemporad, A., & Boyd, S. OSQP: An Operator Splitting Solver for Quadratic Programs. Mathematical Programming Computation 12(4), 2020.

[Stewart1996] Stewart, D. E., & Trinkle, J. C. An Implicit Time-Stepping Scheme for Rigid Body Dynamics with Inelastic Collisions and Coulomb Friction. International Journal for Numerical Methods in Engineering 39(15), 1996.

[Stronge1990] Stronge, W. J. Rigid Body Collisions with Friction. Proceedings of the Royal Society of London A 431, 1990.

[Su2021] Su, J., Lu, Y., Pan, S., Wen, B., & Liu, Y. RoFormer: Enhanced Transformer with Rotary Position Embedding. arXiv:2104.09864, 2021.

[Tobin2017] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., & Abbeel, P. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. IROS 2017.

[Todorov2012] Todorov, E., Erez, T., & Tassa, Y. MuJoCo: A Physics Engine for Model-Based Control. IROS 2012.

[Todorov2004] Todorov, E., & Li, W. A Generalized Iterative LQG Method for Locally-Optimal Feedback Control of Constrained Nonlinear Stochastic Systems. ACC, 2005.

[Tong2023] Tong, A., Malkin, N., Huguet, G., Zhang, Y., Rector-Brooks, J., Fatras, K., Wolf, G., & Bengio, Y. Improving and Generalizing Flow-Matching for Generative Modeling with Optimal Transport. ICML 2023.

[vandenOord2017] van den Oord, A., Vinyals, O., & Kavukcuoglu, K. Neural Discrete Representation Learning (VQ-VAE). NeurIPS 2017.

[Vaswani2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. Attention Is All You Need. NeurIPS 2017.

[Vincent2011] Vincent, P. A Connection between Score Matching and Denoising Autoencoders. Neural Computation 23(7), 2011.

[Walker1982] Walker, M. W., & Orin, D. E. Efficient Dynamic Computer Simulation of Robotic Mechanisms. Journal of Dynamic Systems, Measurement, and Control 104(3), 1982.

[Walke2023] Walke, H. R., Black, K., Zhao, T. Z., et al. BridgeData V2: A Dataset for Robot Learning at Scale. CoRL 2023.

[Westervelt2007] Westervelt, E. R., Grizzle, J. W., Chevallereau, C., Choi, J. H., & Morris, B. Feedback Control of Dynamic Bipedal Robot Locomotion. CRC Press, 2007.

[Xiao2021] Xiao, W., & Belta, C. High-Order Control Barrier Functions. IEEE Trans. Automatic Control 67(7), 2022.

[Yu2020] Yu, T., Finn, C., Xie, A., Dasari, S., Zhang, T., Abbeel, P., & Levine, S. One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning. RSS 2018.

[Yun2020] Yun, C., Bhojanapalli, S., Rawat, A. S., Reddi, S., & Kumar, S. Are Transformers Universal Approximators of Sequence-to-Sequence Functions? ICLR 2020.

[Zhai2023] Zhai, X., Mustafa, B., Kolesnikov, A., & Beyer, L. Sigmoid Loss for Language Image Pre-Training. ICCV 2023.

[Zhao2023] Zhao, T. Z., Kumar, V., Levine, S., & Finn, C. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. RSS 2023.

[Zienkiewicz2005] Zienkiewicz, O. C., & Taylor, R. L. The Finite Element Method (6th edition). Butterworth-Heinemann, 2005.

[Zhou2023] Zhou, Z., Zhu, M., He, C., et al. Spikformer: When Spiking Neural Network Meets Transformer. ICLR 2023.

Riferimenti Aggiuntivi

[Chen2024] Chen, Z., et al. SpatialVLM: Endowing Vision-Language Models with Spatial Reasoning Capabilities. CVPR 2024.

[Dennis2020] Dennis, M., Jaques, N., et al. Emergent Complexity and Zero-Shot Transfer via Unsupervised Environment Design. NeurIPS 2020.

[Dou2024] Dou, Z., et al. GROOT: Learning to Manipulate Novel Objects from Demonstrations. CoRL 2024.

[FigureOpenAI2025] Figure / OpenAI. Helix: A Vision-Language-Motor Model for Full-Body Humanoid Control. Technical Report, 2025.

[GoogleDeepMind2025] Google DeepMind Robotics. Gemini Robotics-ER: Embodied Reasoning for Dexterous Manipulation. Technical Report, 2025.

[Grassberger1983] Grassberger, P., & Procaccia, I. Characterization of Strange Attractors. Physical Review Letters 50(5), 1983.

[Hogan1985] Hogan, N. Impedance Control: An Approach to Manipulation. ASME Journal of Dynamic Systems 107(1), 1985.

[Lake2018] Lake, B. M., & Baroni, M. Generalization Without Systematicity: On the Compositional Skills of Sequence-to-Sequence Recurrent Networks. ICML 2018.

[Locatello2020] Locatello, F., et al. Object-Centric Learning with Slot Attention. NeurIPS 2020.

[Mastalli2022] Mastalli, C., et al. Crocoddyl: An Efficient Framework for Multi-Contact Optimal Control. ICRA 2022.

[Pearl2000] Pearl, J. Causality: Models, Reasoning, and Inference. Cambridge University Press, 2000.

[Rothenberg1971] Rothenberg, T. J. Identification in Parametric Models. Econometrica 39(3), 1971.

[TeamOcto2024] Team Octo. Octo 2: An Open-Source Generalist Robot Policy. arXiv:2405.12213, 2024.

[Wu2024] Wu, Z., et al. DiffusionFEM: Physics-Informed Generative Model for Deformable Object Simulation. ICLR 2024.

[Yuan2024] Yuan, H., et al. RoboPoint: A Vision-Language Model for Spatial Affordance Prediction for Robotics. arXiv:2406.10721, 2024.

[Zhou2023] Zhou, Z., et al. Spikformer: When Spiking Neural Network Meets Transformer. ICLR 2023.

Indice Analitico

L'indice riporta i concetti principali con il rimando al capitolo e alla sezione della monografia. Le sezioni dell'appendice sono indicate con la lettera corrispondente.

Action chunking Cap. 6.2, Cap. 11

Algoritmo CRBA Cap. 2.5, App. B.2

Algoritmo RNEA Cap. 2.5, App. B.1

Attrito di Coulomb Cap. 4.2, Cap. 4.6

Backpropagation Through Time (BPTT) Cap. 14.3, Cap. 14.4

Camminata bipede, stabilità orbitale Cap. 3.3, Cap. 3.5

Centroidal Momentum Matrix Cap. 1.2, Cap. 1.5, App. B.3

Christoffel, simboli di Cap. 1.3, Cap. 1.4, App. C.2

Codifica posizionale (RoPE, sinusoidale) Cap. 5.3

Complessità campionaria (BC, DAgger, GAIL) Cap. 12.1–12.4

Control Barrier Functions (CBF) Cap. 16

Cross-embodiment Cap. 6.1, Cap. 6.3

Delassus, matrice di Cap. 4.2, Cap. 4.5, App. C.4

Diffusion Policy Cap. 11.1, Cap. 11.3

Domain randomization Cap. 9.2, Cap. 9.5, Cap. 22.2

DPO (Direct Preference Optimization) Cap. 5.4

Dreamer V3 / RSSM Cap. 8.1–8.6

Equazioni di Eulero-Lagrange (floating-base) Cap. 1.1–1.4

Flow matching Cap. 10.2, Cap. 11.2, Cap. 11.4

Gauss, principio di Cap. 4.4

GAIL (Generative Adversarial Imitation Learning) Cap. 12.4

Hamilton-Jacobi-Isaacs (HJI) Cap. 17

Hybrid Zero Dynamics (HZD) Cap. 3.3–3.5

Informazione di Fisher, matrice di Cap. 6.4

Integratore time-stepping Cap. 4.3, Cap. 4.6

Leaky Integrate-and-Fire (LIF) Cap. 13.1, Cap. 14.4

LoRA (Low-Rank Adaptation) Cap. 5.1, Cap. 5.4, Cap. 7

LCP (Linear Complementarity Problem) Cap. 4.2, Cap. 4.5

Loihi 2 Cap. 13, Cap. 15.3

Lyapunov, funzione di (WBC) Cap. 2.4

Meccanica di Hertz Cap. 4.7

MPC centroidale Cap. 2.1

Natural Policy Gradient Cap. 6.4

Newton-Darboux, legge di impatto Cap. 4.3

PDDLStream Cap. 19.3, App. B.6

π_0 (Physical Intelligence) Cap. 6, Cap. 10

Poincaré, mappa di Cap. 3.3, Cap. 3.5

QP gerarchico (HQP) Cap. 2.2, Cap. 2.5, App. B.4

Randomized smoothing Cap. 18.3

RoPE (Rotary Position Embedding) Cap. 5.3

RT-2 / OpenVLA Cap. 5

SayCan Cap. 20.1

Scaling laws (robotica) Cap. 7.3

Schrödinger, ponte di Cap. 10.5

SE(3), errore su Cap. 2.3, Cap. 15.1

Signorini, condizioni di Cap. 4.2, Cap. 4.4

SLIP (Spring-Loaded Inverted Pendulum) Cap. 3.4

STDP Cap. 14.2

System identification Cap. 21

Teorema di Brenier Cap. 10.4

Teorema di Grizzle Cap. 3.3.6

Teorema di Nagumo Cap. 16.1

Trasporto ottimale Cap. 10.3–10.5

Transformer, attenzione multi-testa Cap. 5.3, Cap. 7.1

VQ-VAE Cap. 7.2

Wasserstein, distanza di Cap. 10, Cap. 12.1

Whole-Body Control (WBC) Cap. 2

ZMP (Zero Moment Point) Cap. 3.1, Cap. 16.3